An Approach towards
Context-sensitive and User-adapted
Access to Heterogeneous Data Sources,
Illustrated in the Television Domain

Pieter Bellekens
Bellekens, Pieter

An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain / by Pieter Bellekens.

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-2336-8
NUR 983

Keywords: Information integration / user modeling / data retrieval / personalization / interactive television
An Approach towards
Context-sensitive and User-adapted
Access to Heterogeneous Data Sources,
Illustrated in the Television Domain

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven,
op gezag van de rector magnificus, prof.dr.ir. C.J. van Duijn,
voor een commissie aangewezen door het College voor Promoties
in het openbaar te verdedigen
op donderdag 7 oktober 2010 om 16.00 uur

door

Pieter Alfons Elisabeth Bellekens

geboren te Mechelen, België
Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. P.M.E. De Bra
en
prof.dr.ir. G.J.P.M Houben

Copromotor:
dr. L.M. Aroyo
Acknowledgments

Back in 2005 when dr. Lora Aroyo first proposed me a PhD position in the ITEA Passepartout project, I immediately liked the prospect of four more university years within a more professional research setting. However, I would soon discover that doing a PhD is more than just working fulltime on one and the same project. It is a long term commitment which consumes a lot of energy, effort and dedication, including many hard lessons to learn. No pain, no gain. Luckily, the pleasure and profit from fruitful collaborations, discussing with kindred spirits, working in an international project, doing research with cutting edge technologies, etc., largely outweighs any difficulty or setback. Now five years later, I look back with pride and enjoyment to the accomplishments in terms of work, but also to my personal evolution during the course of these years.

First, I would like to express my gratitude to my supervisors prof. Paul De Bra, prof. Geert-Jan Houben and dr. Lora Aroyo. I am grateful to Paul De Bra for his valuable guidance throughout the entire course of my PhD and for the creation of a nice unrestrained research environment. Mostly, I will remember his open and positive mood which led to many discussions concerning both work and our shared hobby, photography. I am thankful to Geert-Jan Houben for his endless stream of valuable ideas, input and remarks. Often, he provided the keys to open a closed door and solved problems through a multitude of discussions and reflections. Further, I would especially like to thank Lora Aroyo with whom I started this project. She was always just around the corner for any discussion, and provided all possible facilities and connections to support my research with an unprecedented enthusiasm and passion for the field.

I also would like to thank all the people I collaborated with during my research. Especially my colleagues Kees van der Sluijs, Martin Björkman and Peter Barna with whom I shared an office, which resulted in friendship and several joined publications. I would like to thank Ad Aerts, Philippe Thiran and Jeen Broekstra for their valuable input during their stay at our faculty. Further, I would like to thank Toon Calders and Philipp Cimiano for their expertise in the field of data mining and knowledge representation respectively. Many thanks to my master students Tim Dekker, Erik Loef, Charanjeev Kaur, Chris Smeets, Roland Schijvenaars and Jan van Nunen for their participation in the project and their different contributions in terms of software development and model generation. I also would like to thank Reinier Post and Eric Verbeek for solving many technical server and database related issues. Lastly, I am grateful to all my colleagues at the Information Systems group and specifically our secretaries Riet van Buul and Ine van der Ligt for their help involving any administrative task.

A special word of thanks goes to Annelies Kaptein, CEO of Stoneroos, and the whole Stoneroos team with whom I had the pleasure to work with during and after my PhD. Annelies, thank you very much for all the support and understanding.

Further, I would like to thank all my dear friends, both in the Netherlands and Belgium, for their continuous support, encouragement, listening and provisioning of fun! Richard, thanks for proof-reading my introduction. Erik, thanks for the cover design!

Lastly, I would like to express my gratitude to my parents and my sister for their continuous and unconditional support. It is thanks to them that I was able to achieve everything I accomplished so far. Their belief and support was what I needed to push through and complete this dissertation.
Contents

1 Introduction ........................................ 1
   1.1 Motivation ........................................ 2
   1.2 State of the art .................................. 4
   1.3 Research Questions .............................. 5
   1.4 Outline and contributions .................... 7

2 The Television Domain ............................. 11
   2.1 A little bit of history... ..................... 11
   2.2 Interactive television applications .......... 13
      2.2.1 Electronic Program Guide ............... 15
   2.3 Set-top box hardware .......................... 16
   2.4 Television 2.0 .................................. 17
      2.4.1 Television user behavior .................. 18
      2.4.2 Looking at the future ..................... 18
   2.5 Modeling the domain .......................... 20
      2.5.1 Metadata: a definition ..................... 21
      2.5.2 State-of-the-art: Audiovisual Description schemas .. 21
      2.5.3 Comparison ................................ 36
      2.5.4 Conclusion ................................ 39
   2.6 Conclusion ..................................... 40

3 Requirements ..................................... 43
   3.1 Introduction ..................................... 43
      3.1.1 Some definitions ............................ 44
   3.2 Requirements ................................... 45
      3.2.1 Domain model ............................... 45
      3.2.2 User model ................................ 46
      3.2.3 Adaptation ................................ 47
   3.3 Choosing the data model ....................... 48
      3.3.1 Escort 2.4 .................................. 48
      3.3.2 XML-TV .................................... 49
      3.3.3 LSCOM ..................................... 50
      3.3.4 TV-Anytime ................................ 50
      3.3.5 ProgramGuideML ............................ 51
      3.3.6 Conclusions ................................ 52
   3.4 TV-Anytime ..................................... 52
      3.4.1 Introduction ................................ 53
      3.4.2 TV-Anytime Phase I ....................... 54
      3.4.3 TV-Anytime Phase II ....................... 58
   3.5 The Semantic Web ............................... 61
      3.5.1 XML's Shortcomings ....................... 61
      3.5.2 A Web with meaning ....................... 63
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6</td>
<td>Conclusion</td>
<td>66</td>
</tr>
<tr>
<td>4</td>
<td>An Approach to Adapt Access to Heterogeneous Data</td>
<td>67</td>
</tr>
<tr>
<td>4.1</td>
<td>A General Approach</td>
<td>67</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Integration</td>
<td>69</td>
</tr>
<tr>
<td>4.1.2</td>
<td>User Modeling</td>
<td>69</td>
</tr>
<tr>
<td>4.1.3</td>
<td>User-adapted data access</td>
<td>70</td>
</tr>
<tr>
<td>4.2</td>
<td>Conclusion</td>
<td>71</td>
</tr>
<tr>
<td>5</td>
<td>Integration</td>
<td>73</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>73</td>
</tr>
<tr>
<td>5.2</td>
<td>TV-Anytime in RDF(S)/OWL</td>
<td>74</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Metadata Schemas</td>
<td>74</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Classification Schemes</td>
<td>77</td>
</tr>
<tr>
<td>5.3</td>
<td>Schema Integration</td>
<td>79</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Addition of temporal and spatial semantics</td>
<td>79</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Addition of Lexical semantics</td>
<td>85</td>
</tr>
<tr>
<td>5.3.3</td>
<td>SKOS enrichment of Classification Schemes</td>
<td>86</td>
</tr>
<tr>
<td>5.3.4</td>
<td>The iFanzy Domain Model Overview</td>
<td>89</td>
</tr>
<tr>
<td>5.4</td>
<td>Instance data integration</td>
<td>90</td>
</tr>
<tr>
<td>5.4.1</td>
<td>BBC Backstage</td>
<td>90</td>
</tr>
<tr>
<td>5.4.2</td>
<td>XML-TV</td>
<td>93</td>
</tr>
<tr>
<td>5.4.3</td>
<td>IMDb</td>
<td>96</td>
</tr>
<tr>
<td>5.4.4</td>
<td>VideoDetective</td>
<td>100</td>
</tr>
<tr>
<td>5.4.5</td>
<td>DBpedia</td>
<td>101</td>
</tr>
<tr>
<td>5.5</td>
<td>Overview</td>
<td>102</td>
</tr>
<tr>
<td>5.6</td>
<td>Conclusion</td>
<td>105</td>
</tr>
<tr>
<td>5.7</td>
<td>Related Work</td>
<td>106</td>
</tr>
<tr>
<td>5.7.1</td>
<td>State-of-the-art applications</td>
<td>106</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Alignment and integration techniques</td>
<td>107</td>
</tr>
<tr>
<td>6</td>
<td>User Modeling</td>
<td>109</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>110</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Explicit vs. Implicit</td>
<td>111</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Context</td>
<td>112</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Semantic Web Technologies</td>
<td>113</td>
</tr>
<tr>
<td>6.2</td>
<td>The User Model</td>
<td>114</td>
</tr>
<tr>
<td>6.2.1</td>
<td>The User Model Structure</td>
<td>114</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Context-sensitive User Generated Events</td>
<td>116</td>
</tr>
<tr>
<td>6.2.3</td>
<td>User likings</td>
<td>119</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Mining User Event Patterns</td>
<td>121</td>
</tr>
<tr>
<td>6.3</td>
<td>Cold Start</td>
<td>128</td>
</tr>
<tr>
<td>6.3.1</td>
<td>User watching statistics</td>
<td>129</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Community profiles</td>
<td>131</td>
</tr>
<tr>
<td>6.3.3</td>
<td>User classifications</td>
<td>134</td>
</tr>
<tr>
<td>6.4</td>
<td>Conclusion</td>
<td>135</td>
</tr>
<tr>
<td>6.5</td>
<td>Related Work</td>
<td>136</td>
</tr>
<tr>
<td>7</td>
<td>User-Adapted Data Access</td>
<td>139</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>140</td>
</tr>
<tr>
<td>7.2</td>
<td>A personalized data access strategy</td>
<td>142</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Query Refinement</td>
<td>143</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Database Management System</td>
<td>145</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Content Filtering</td>
<td>146</td>
</tr>
</tbody>
</table>
CONTENTS

7.2.4 Recommender System ............................................. 148
7.3 Conclusion .......................................................... 152
7.4 Related Work ....................................................... 152
  7.4.1 The broad field of personalization ......................... 152
  7.4.2 Personalizing data access .................................. 154

8 Data Retrieval Performance ........................................... 157
  8.1 Introduction ....................................................... 158
  8.2 Query Optimization and Performance ......................... 158
    8.2.1 A naive first approach ................................... 159
    8.2.2 Decomposition in Sources and Querying ............... 160
    8.2.3 Applying available tools, technologies and techniques ........................................ 163
  8.3 Conclusions ....................................................... 165

9 Interfaces and Interaction ............................................ 167
  9.1 Introduction ....................................................... 168
  9.2 The back-end server ............................................. 168
    9.2.1 Content Retrieval Layer ................................... 170
    9.2.2 Querying and Clustering Layer ......................... 170
    9.2.3 Personalization Layer .................................... 171
    9.2.4 Server Administration Layer ........................... 171
  9.3 SenSee ............................................................ 172
  9.4 The iFanzy user interfaces ..................................... 175
    9.4.1 The iFanzy Web site ....................................... 175
    9.4.2 The set-top box interface ............................... 177
    9.4.3 The iPhone application .................................. 179
    9.4.4 iFanzy interface evaluation ........................... 180
  9.5 Conclusion ....................................................... 185
  9.6 Related Work ..................................................... 185

10 Evaluation .......................................................... 189
  10.1 Recommendation evaluation .................................... 189
    10.1.1 Goals ...................................................... 190
    10.1.2 Setting .................................................... 191
    10.1.3 Results .................................................... 193
    10.1.4 Comparison ............................................... 199
    10.1.5 Discussion ............................................... 200
  10.2 Increasing Serendipity ......................................... 201
    10.2.1 Explicit Semantic Analysis ............................ 202
    10.2.2 Integrating ESA into the iFanzy recommender system ........................................ 202
    10.2.3 Evaluation ............................................... 203
  10.3 Conclusion ....................................................... 206
  10.4 Related Work ..................................................... 206

11 Conclusions and Future Work ....................................... 209
  11.1 Conclusions ..................................................... 209
  11.2 Future Work ..................................................... 216
    11.2.1 Integration ............................................... 216
    11.2.2 User modeling .......................................... 216
    11.2.3 User-Adapted Data Access ............................. 217
    11.2.4 The future of television ............................... 217
## A Data sources

A.1 XML-TV .................................................. 219  
A.1.1 XML-TV DTD ........................................... 219  
A.1.2 XML-TV RDFS Schema ................................. 222  
A.2 IMDb ..................................................... 223  
A.2.1 IMDb Title RDFS Schema ............................... 223  
A.2.2 IMDb Person RDFS Schema .............................. 224  
A.2.3 IMDb Literature RDFS Schema ......................... 225  
A.2.4 IMDb Company RDFS Schema ........................... 225  
A.2.5 IMDb Location RDFS Schema ........................... 225  
A.2.6 IMDb Photo RDFS Schema .............................. 226  
A.2.7 IMDb Trailer RDFS Schema .............................. 226  
A.2.8 IMDb Episode RDFS Schema ............................ 226  
A.3 VideoDetective .......................................... 227

Bibliography .................................................. 227  
List of figures .................................................. 245  
List of tables .................................................. 248  
Glossary ....................................................... 249  
Summary ....................................................... 251  
Samenvatting .................................................. 254  
Curriculum Vitae ............................................. 256
Chapter 1

Introduction

With the rise of new technological advancements seen in recent years, people's lives have changed considerably. New technologies sprouted in many different fields and led for example to smart handheld devices which allow people to search and browse the Web at all times, the convergence of users forming various social networks and communities, sources like libraries, shopping malls, airports, etc. are opening their databases to the public, an incredible increase in data throughput introducing vast amounts of new audiovisual media and last but not least many more users with the ability to access this global network of information.

However, while new technologies should primarily be intended to make life easier, in many cases the application of new technologies has led to increasingly complicated tasks and requiring more effort from the user. Take for example the evolution of airports, shops, libraries, etc. opening up their databases by means of new technologies like *RESTful Web services*\(^1\) in combination with *JSON*\(^2\) (used for serializing and transmitting structured data over a network connection). Many third-party Web sites started gathering information from those sources to present it to the user in a unified manner. E.g. a portal to search for the best flight between \(A\) and \(B\), the cheapest price for product \(X\) across all stores, etc. While this was undoubtedly a positive trend, after some time many of these Web sites started to get biased and presented information in contradicting ways by for example favoring specific companies, giving special promotions, etc. This left users confused, eventually putting the burden of finding the optimal flight or cheapest price back on their shoulders. In a different domain, new technologies like digital television, Web-connected set-top boxes and High Definition media have also influenced the user. At home, people found themselves, after purchasing a brand new high-tech digital television set-top box, zapping through hundreds of television channels hoping to find something of interest for them. In the end, most of them will, slightly disappointed, return to watch the ten channels they always watched before. Another case involves the digitalization of radio broadcasts which enlarged the number of available channels from tens to tens of thousands, where the user himself now painstakingly has to find the channel fitting him best. Or take RSS technology, which can keep us up-to-date about any subject at all times, by sending a ping when a new message has arrived. With only a few subscriptions, people are constantly spammed with messages, while usually only a fraction is considered interesting enough to read. Powerful search technologies, which once were our search companions, can now return millions of allegedly relevant results in an attempt to answer a simple question. Generally, in each of these cases the user needs to decide which channel to listen to, which news item to read, which search result to investigate further, which TV program to watch, etc., again making the user responsible to find the best pick. On top of this, because of the technological leaps in the capabilities of handheld devices, we are constantly connected to the Internet and continuously have to make such decisions until we ourselves get overloaded by the huge and never-ending information provisioning.

\(^1\)[http://en.wikipedia.org/wiki/Restful]
\(^2\)[http://en.wikipedia.org/wiki/JSON]
CHAPTER 1. INTRODUCTION

As often, the main underlying problem in all of these situations is choice. People need to choose a TV program, a radio station, a book, a Web site, a news item, a friend, etc. out of an increasingly large world of possibilities. As previously adequately put, at some point, choice, often referred to as the hallmark of individual freedom and self-determination, becomes detrimental to our psychological and emotional well-being [212]. Basically, too much choice inevitably leads to less satisfaction with the final choice, possibly inducing an aversion against the technologies providing it. Still, the main purpose of the technologies listed above is to try to present as much choice as possible. Moreover, for digital television broadcasters, radio channel providers, search engines, etc., their huge collections of items is a selling point. The more the better! Unfortunately, they, along with most others, believe that the user himself benefits from a large range of options and that he or she is capable enough to make the right choice.

Luckily, most people still think of all these technological evolutions as a good thing. However, the challenge of the future is to help people finding their way through the forest of choice. Instead of providing more options, possibilities and features, we need to introduce a smart pre-selection step which decreases the choice without losing the pick of the bunch. Ideally, such a system could revolutionize our lives by presenting exactly those television programs you want to watch, that radio station you want to listen to, the Web site you want to read or even the clothes you want to wear, the new car you want to buy in the colors of your liking or the food you want to eat. Applications making those difficult choices for us or at least support us in making them, can help us to again enjoy the choices made.

In Section 1.1 of this chapter, we provide the motivation for this research and give a brief overview of our approach. Afterwards in Section 1.2, we provide a short introduction to the state-of-the-art in the television domain. Later in Section 1.3, we define the various research questions addressed, while Section 1.4 shows the relation between the research questions, chapters and different research contributions.

1.1 Motivation

Over the past ten years, we all witnessed the incredible evolution and success of the World Wide Web. Both the number of users and data grew and still grows enormously. However, this unbridled growth also leads to the situation where it becomes increasingly difficult to deal with these large amounts of content, often spread heterogeneously, and serve it intelligently to a huge number of users. No doubt, when the haystack grows, finding the needle becomes increasingly harder too. However from a different perspective, users today are also expecting more and more from data retrieval engines. Not just any needle will do anymore. For example, after the introduction of other Internet capable devices besides a computer like for example a PDA, smart phones, tablet computers, etc., users became more aware of the possibilities and demanding towards data retrieval. After all, people nowadays expect different behavior and different results specifically chosen to match their current device and situation. The more the Web had to offer, the more fastidious people became.

At the start of this dissertation project, surprisingly, one of the last strongholds of electronic devices not connected to the Internet, was the television platform. Surprisingly indeed, because the television is still one of the technologies with the highest global penetration rate. The fact that televisions remained dumb stream-playing devices, is mainly due to the fact that it proved hard to come up with a standardized technology which could connect every TV set to the Internet and allow for the rather large and necessary data throughput. Hence, it requires a personal video data stream as opposed to generally employed broadcast, which is the same for all receivers. However, with recent technological advancements it seems that finally also the television platform will slowly evolve towards the next Internet gateway. The platform is gaining interest expeditiously, considering its new possibilities like Video-On-Demand, Electronic Program Guides, various Web channels, online TV games, etc.

The result of this integration will be that many new users will be introduced in the online world, hundreds of millions of existing users will change and/or increase their Web consumption
behavior and a huge amount of new audiovisual content will become available due to sharing of
user-generated content and more programs being made available on demand. Again, more and
more content providers will start struggling with rapidly growing content sets and users will set
more exotic demands to how they would like their data to be served with a minimum of effort
from their side. In essence, there is a general need for a structured approach which allows for
the integration of huge data sets, takes care of the user’s personal preferences, interests, etc. and
allows the user to access this data in a user-adapted and intuitive way.

Assuming that the amount of available information on the Web is going to grow continuously,
making the right choice will, without any help, become proportionally difficult. In this dissertation,
we investigate the definition of an approach which can adapt the data retrieval process to help
the user in finding the best available data in any specific situation. Furthermore, this approach
should not depend on the domain and thus be applicable in any domain involved in the retrieval
of data (like for example books, TV programs, radio channels, news items, etc.). In such an
approach we foresee three essential parts each contributing towards the solution of the problem:
i) the modeling of key items in the domain and the integration of their metadata, from different
sources, ii) modeling all aspects of the relevant user(s) and the integration into one exhaustive
user model and iii) a strategy to compare the data set on the one hand and the user’s perspective
on the other, enabling a smart selection of, for that user, interesting items.

The integration of data is important because different metadata sources can be heterogeneous
i.e. different in structure and/or syntax. Having a lot of heterogeneous data describing one par-
ticular item is nice, however it becomes more useful when properly combined into one consistent
structure. This integration will therefore facilitate that every item in the domain (e.g. books, CDs,
programs, etc.) is described as richly as possible, given the available data sources. However, this
integration requires a profound metadata structure in which this heterogeneous data can be stored.
Such a structure should at least provide a well-defined semantic structure, support reasoning over
its data to deduce new statements, provide a wide diversity of properties, allow the identifiability
of every item and be able to uniquely reference other external resources.

The second part concerns the modeling of the user. Such a user model will maintain all the
different facets of the description of that user, such that the retrieval process can be tailored to
match the user’s needs. After all, any kind of personal data processing requires some form of
structured user information. However, we do not want to bother the user too much with the
creation of such a model. Therefore, the model must consider, besides direct feedback requiring
the user’s attention explicitly, also the user’s behavior obtained by monitoring him of her implicitly
and unobtrusively. From this feedback, we can then extract the user’s perspective on different
domain items. However, the user’s view on these items is a very situational matter. People like
different things in the morning, when they are with friends, when they feel sad, etc. This means
that at any time it is important to know in which specific circumstances particular user feedback
holds. Therefore, a user model should be aware of the context in which all user feedback was valid.
Lastly, application which try to accommodate users in a personalized way, experience problems
whenever new users, who still have an empty user model, subscribe to the system. Therefore,
the user model is responsible to gather and integrate as much user data as possible, even from
different sources if available, in an effort to alleviate this problem.

The third and last part involves the data retrieval strategy itself, exploiting both the metadata
of the domain items and the description of the user. Given a set of well-described domain items and
a model of user $X$, this strategy will predict which items will be enjoyed most by $X$ at any specific
point in time. This strategy should be generic and flexible enough to facilitate a personalized
content retrieval service employable in both searching and recommendation algorithms. In essence,
this approach chooses the best available items, relieving the user from making these choices.

To demonstrate and evaluate the proposed approach, we illustrate it in the television domain
which is currently still in its infancy with respect to smart data retrieval. As previously described,
the TV domain is about to be connected to the Web which will suddenly enable a whole new
world of interactivity to potentially more than 1.2 billion households worldwide$^3$.

$^3$Statistics from the International Television Expert Group in 2010 (http://www.international-
1.2 State of the art

The approach described in this thesis is a general recipe to enable user-adapted services in user-centered domains. Of this, the television domain is a good example. People like watching programs, discuss them with friends, rate, record and archive them, create favorite actors, directors, etc. However, considering the evolution the personal computer in combination with the Internet has gone through, television systems suffer from a great backlog. This is mainly due to the fact that the television domain is a very conservative one. Why would a content producer or a broadcaster for example invest in High Definition cameras and infrastructure if no-one has a HD television? However at the start of this dissertation, we could see that more and more companies started to invest and research the possibilities to enlarge the scope of the television platform in terms of interactivity and connectivity.

Considering that a television is just a dumb device to show images without any intelligence, connectivity or data awareness whatsoever, a set-top box was introduced to provide new functionalities which could be drawn from the input signal and pushed to the TV screen. However, initially it proved difficult to provide real interactivity because of the lack of a return channel (the ability to send data back to a server), disallowing 2-way communication. All data (including “interactive services”) was pushed to the set-top box which further dealt with this content locally. Due to this highly constrained setting, possibilities were limited. It was impossible to register the user’s context, behavior, preferences, reflection on the programs, etc. At best, such information could only be maintained and parsed locally on the box, which was usually a very slow device.

Acknowledging these problems, the market introduced a new generation of set-top boxes, which included an Internet connection, providing the anticipated return channel and thereby opening a whole new world of possibilities. Through this evolution, set-top boxes nowadays allow the users for example to watch content on-demand, check their emails or fetch specific Web content. However, in comparison to modern day Web applications, the television platform still has a long way to go, particularly in terms of content retrieval, personalization, navigation, interactivity, etc. Moreover, having witnessed the growth of content on the Web, we can safely assume that the same will happen in the television domain. When the amounts of available content start to rise exponentially, zapping through channels will soon become inadequate and a personalized user-centered approach will be required to help people in finding what they are looking for.

Unfortunately, there are currently not many approaches available to deal with this evolution, for example by means of personal search or context-sensitive recommendations. TiVo, one of the most advanced and successful commercial attempts to bring a new generation of Digital Video Recorder (DVR) to the market [21], employs one approach to content recommendation. By means of a simple “thumbs up, thumbs down” strategy, it gathers the user’s opinion, and afterwards compares this feedback to different TiVo users. By using collaborative filtering this comparison is then used for the recommendation of other programs. However, this approach is pretty limited since it for example does not take the user’s context into account and only focusses on the television platform, ignoring any other device. The user data they maintain is limited to the direct “thumbs up, thumbs down” feedback, which is hardly sufficient to provide personalized and context-sensitive access to data in a ubiquitous environment.

To facilitate such advanced data retrieval functionalities, a good knowledge of the domain items is indispensable. Usually, a domain and its instances are described by some form of metadata specification or domain model which outlines the semantics of the objects and their relationships. Within the television domain many different metadata specifications have been defined, each with its specific goal, advantages and disadvantages. However, many of these specifications are primarily built for B2B (Business-2-Business) purposes, and the ones which are not are often limited in terms of expressiveness, richness and flexibility. The best available candidate, to serve as basis for a rich description of television programs, is the TV-Anytime specification [238]. TV-Anytime is a schema which was specifically made by a consortium of television related companies for next generation television systems. Despite of a few minor issues, TV-Anytime contains constructs to express

http://television.org/tv_market_data/world-tv-market-2010.html
1.3. RESEARCH QUESTIONS

almost every possible feature of a television program in considerable depth. Further, it provides the necessary constructs to richly and completely describe any given television program, and can uniquely identify any program by means of a CRID. In reality however, TV-Anytime is rarely used. Moreover, most existing approaches use proprietary formats or only use parts of existing specifications. This is probably due to the fact that current set-top box applications do not have the need for rich metadata descriptions. Hence, TV programs in such applications are often poorly annotated, just containing a title, synopsis and one or more genres. However, in order to create personalization strategies, a rich metadata description is paramount. Hence, the more you know about a program, the better you can predict how strongly it matches the user.

Looking at the current state-of-the-art, we can see that available set-top box applications are far from ready to deal with upcoming evolutions in the field. Moreover, there is also no strategy or approach devised to apply when the need of smarter data retrieval arises.

For a more elaborated view on both the current state-of-the-art of the television domain and requirements for our approach, we would like to refer to Chapter 2 and 3 respectively. In the following section we list the different research questions following from both our motivation and the limitations of available systems.

1.3 Research Questions

In the previous sections we identified that the main problem for the user is coping with the upcoming abundance of choice. Therefore, we introduced an approach which is devised to support the user in making that choice. This approach facilitates user-adapted data access which means that whenever the user wants to search for items, this approach will aid in filtering and selecting the available results such that the best results, for this user in this situation, are proposed.

In this section we formulate nine research questions which we address in this dissertation. The answers to these questions contribute to the definition, state-of-the-art, implementation and evaluation of the approach aimed at adapting the data retrieval process. The research questions addressed in this dissertation include:

Research Question 1

Which technologies and standards, prevalent in the television domain, exist and can be suitable to support interactive and personalized television applications?

To illustrate our approach in the television domain, some background information of the TV domain is required. We take a look at the state-of-the-art, and investigate which types of applications, software, hardware, etc. currently exist. With the previously discussed approach in mind, we take a closer look at different standards and more importantly, existing metadata schemes describing television programs.

Research Question 2

What are the requirements for an approach providing user-adapted data retrieval?

Here, we consider the requirements for the semantic structure of the domain or domain model, the semantic structure of the user model and the data retrieval adaptation strategy. Knowing which technologies and standards exist in the television domain, we can investigate in how far they can serve these requirements and where they are insufficient.

Research Question 3

Which generic approach has the potential to provide user-adapted data access to large heterogeneous data sources?

Previously, we indicated that an approach towards user-adapted data retrieval roughly consists of three important parts. Through the previously defined requirements of such an approach, we
can deduce how these parts should fit together and which inputs they require from one another. Next, we can draw up a matching architecture.

Research Question 4

_How can we integrate large heterogeneous data sources into one consistent and semantically rich data model?_

The first major part in the approach previously discussed, involves the integration of data from different heterogeneous sources. This data integration is important since different sources can contribute different perspectives on the same resources. To facilitate this integration, we rely among others on techniques from the Semantic Web often used to overcome heterogeneities between different metadata schemes.

Research Question 5

A: _How can we model relevant user data to support context-sensitive adaptation?_

B: _How can we obtain this user data encompassing both explicit and implicit data?_

C: _How can we support new users who suffer from an empty user model?_

The second part in the approach involves modeling the user. A user model comprehends all relevant information, which can be elicited from that user. Considering the amount of both explicit and implicit data that a user potentially can generate, we need to investigate measures to filter relevant patterns and consolidate them into a strong but representative model. However, problems can arise when a new user, still lacking a user model, enters the system. To support those, we investigate a number of methods to alleviate this problem.

Research Question 6

_How can we provide user-adapted data access given a well-defined domain model and a comprehensive user model?_

Having both a domain model and user model following the requirements, we need to consider how we can exploit this knowledge to provide user-adapted access to that data. This involves researching every step in the data retrieval process, and looking at how each step can be adapted to the user. Further, we also need to investigate the similarities and differences of different types of data retrieval, including for example user-driven search and system-driven recommendations.

Research Question 7

_Given our approach towards user-adapted data retrieval, how can we optimize data retrieval efficiency in terms of querying speed?_

The need for optimization comes naturally with growing amounts of data, numbers of users, requests, integrations, etc. Therefore, we investigate different optimization techniques, like decompositions of data, the creation of various indices, etc., which can increase data retrieval performance. While such optimizations originated from relational database research, here we apply them on RDF repositories. Therefore, considering that these optimizations also lead to the splitting of queries and the joining of partial results, we need to investigate proper utilization of these techniques not to compromise the final query result.

Research Question 8

_How can the user interact with such a system, following our proposed approach and applied in the television domain, effectively?_

The techniques and algorithms applied in this approach are not always straightforward to understand at first glance. Moreover, this complexities should not be visible to the user at all. Therefore, we investigate how the user can interact and benefit from our approach fully, unobtru-
sively and without knowing what works behind the scenes. Furthermore, we investigate different ways for the user to obtain valuable information in a user-friendly manner.

**Research Question 9**

A: What characteristics of recommendations are perceived by users as important for determining the quality of a recommendation?

B: How can we measure the quality of recommendations generated by our approach?

An important aspect of data retrieval in our approach, is to generate recommendations. Therefore, we need to investigate which characteristics are perceived by the users as important, and how we can evaluate the quality of recommendations based on these characteristics.

Although we identified a rather large set of research questions, we have to point out that the core of this thesis revolves around research question 4, 5 and 6. These three questions together formulate an approach which models the domain, models the relevant user properties and afterwards uses both to provide user-adapted access to that data. The other research questions are relevant within this context, to describe the domain and requirements on the one hand and the evaluation in terms of data storage, interfaces and recommendations on the other.

### 1.4 Outline and contributions

In Chapter 2 we deal with research question 1. In this chapter, we look at how different types of digital television applications were constructed and designed at the start of this thesis. We further contemplate how the future digital television platform could fit among other existing platforms like a mobile phone and a desktop computer. This research reflects back on previous work presented in [13], co-authored by Lora Aroyo, Martin Björkman, Geert-Jan Houben, Paul Akkermans and Annelies Kaptein. We conclude this chapter with an extensive overview of existing metadata specifications for the television domain.

Chapter 3 addresses research question 2 by setting domain-independent requirements for an approach to provide user-adapted access to large heterogeneous data sources. More specifically, it emphasizes on the requirements of the domain model, the user model and an adaptation strategy, illustrated with examples from the television domain. These requirements were inspired by requirements previously defined in [13]. Given the data model requirements, we contemplate on the metadata specifications listed in Chapter 2, and find TV-Anytime to be the best candidate if represented in languages supported by the Semantic Web.

In Chapter 4 we propose our approach designed to support user-adapted data retrieval. The approach consists of three main parts in which the first models and integrates data from various heterogeneous sources by means of Semantic Web techniques. The second tries to model the user, maintaining his or her relevant characteristics, preferences, interests, etc., while every statement reflects back on resources defined in the data model. The third and final part enables the generation of user-adapted access given this domain and user model. The presented approach provides an answer to research question 3 and was inspired by the first approach towards a personalized home media center, presented in [40] and co-authored by Martin Björkman, Lora Aroyo, Tim Dekker, Erik Loef and Rop Pulles.

Chapter 5 addresses research question 4, explaining the integration of heterogeneous data as introduced in Chapter 4. It shows, by means of examples from the TV-Anytime specification, why the integration of data from different sources is useful and how it can be facilitated. Afterwards, we again turn to the television domain and show which sources can be used to retrieve and enrich television program metadata. Chapter 5 is based on research previously published in [24, 12], co-authored by Lora Aroyo, Geert-Jan Houben, Annelies Kaptein, Martin Björkman and
Kees van der Sluijs. Parts of this work also appeared in [216]. In [11], co-authored by Lora Aroyo, Geert-Jan Houben, Jeen Broekstra and Martin Björkman, we show that integration of data is also valuable in other domains like for example the Cultural Heritage domain.

Chapter 6 addresses research question 5, which deals with user modeling as the second part in our general approach. In this chapter, we propose a framework which keeps track of all the relevant user’s characteristics, preferences, interests, etc. by monitoring his or her behavior. Every action, both explicit and implicit, produced by the user is kept in a context-sensitive event model. Subsequently, from this model all relevant data is filtered and materialized in the consolidated user model. Further, Chapter 6 introduces a set of measures to alleviate the cold start problem which occurs when new users, with an empty user model, enter. Chapter 6 is based on research previously published in [25], co-authored by Lora Aroyo, Geert-Jan Houben, Annelies Kaptein and Krijn Schaap.

Chapter 7 addresses research question 6, proposing a strategy facilitating user-adapted data retrieval as the third part of the approach presented in Chapter 4. In this chapter, we devise a strategy we call the “adaptation loop”. In this loop, every user request follows a number of adaptation steps before being sent to the database. When matching results are returned, the loop continues to further adapt these results, and finally sends them to the user, closing the loop. The research described in this chapter is based on research presented in [22], co-authored by Lora Aroyo and Geert-Jan Houben.

In Chapter 8, we address research question 7 which deals with the performance of the proposed data retrieval strategy. Here, we try to improve data retrieval efficiency, in terms of querying speed, by means of a number of optimization steps. To illustrate the improvement, we start with a naive implementation of one big data set where no optimizations are applied. Then, we stepwise introduce optimization techniques, like for example keyword indices, decomposition of data sets, etc., gradually improving the performance. The work in this chapter was previously published in [29], co-authored by Kees van der Sluijs, William van Woensel, Sven Casteleyn and Geert-Jan Houben.

Chapter 9 provides an answer to research question 8, dealing with user interaction and interfaces. In this chapter we illustrate two different client applications running on top of a server implementation of the approach devised in Chapter 4. The first client application, which is called SenSee, involves a scientific prototype which we presented at [28] and later at the Semantic Web Challenge [24], co-authored by Lora Aroyo, Geert-Jan Houben, Annelies Kaptein and Kees van der Sluijs. The second interface involves iFanzy, which was and still is developed by Stoneroos, and currently runs on a set-top box, on the iPhone and as a Web portal. To arrive at this point, many different successive versions of the interface were developed and presented in [6, 26, 27, 23], co-authored by Paul Akkermans, Lora Aroyo, Kees van der Sluijs, Geert-Jan Houben and Annelies Kaptein. Lastly, many students of the course Men-Machine-Interaction tutored by Paul de Bra, contributed in the evaluation of the iFanzy Web interface.

Chapter 10 addresses research question 9, evaluating the performance of the recommendation engine. In this chapter we introduce the results from a user study carried out by 60 participants who used and evaluated recommendations presented within the iFanzy Web interface. A condensed version of the results of this test was earlier published in [25]. This chapter is further partially based on research in [241]. Later, the recommendation engine was extended by exploiting “semantic relatedness” between items to increase the serendipity of the system. This work was done in close collaboration with Geert-Jan Houben and Philipp Cimiano. To validate this technique, we reuse results from the initial user study to quantify the increase of surprising results.

Chapter 11 concludes this dissertation with an overview of both conclusions, reflections and suggestions for future work.
Lastly, independent of specific research questions, we would like to stress the contribution of this project as a whole. The main contribution comes from the creation of a domain-independent approach or methodology which can provide user-adapted access to information, given a well-structured set of data and a comprehensive user model. Within the context of the ITEA Passepartout project, in a strong collaboration with Stoneroos Interactive Television\(^4\), we developed iFanzy, a Personalized Electronic Program Guide, following exactly this approach. iFanzy, which in the meantime grew into a commercial product, is currently aimed at introducing some of the next-generation features into the previously closed television world, supporting the television viewers in making the right choices with respect to choosing which channels/content to consume.

\(^4\)http://www.stoneroos.nl/
Chapter 2

The Television Domain

The exponential growth of the Internet we all witnessed during the last decade was and still is remarkable. The amount of information at the disposal of each person has risen exponentially and the Web can now be regarded as the most ubiquitous information source around. Libraries are exposing their contents online, people are sharing their pictures and movies, community-driven encyclopedias and dictionaries are growing both in number and content, search engines can barely keep up with the number of new Web sites created every day, and then we are not even talking about the so-called deep Web [30]. If we compare this steep evolution of the Internet medium, now available on almost any device, with the more conservative broadcast medium on the television platform, the difference is striking. In this chapter we take a closer look at the state-of-the-art of the television domain, providing the necessary insights relevant in the rest of this dissertation and answering research question 1 (Which technologies and standards, prevalent in the television domain, exist and can be suitable to support interactive and personalized television applications?).

2.1 A little bit of history...

The Internet, the global network of interconnected networks we are currently all so familiar with, is, despite its current size, still a relatively recent invention. It evolved from the so-called ARPANET, the world’s first packet switching network, initially created by the Defense Advanced Research Projects Agency (DARPA) of the United States Department of Defense in 1968. The World Wide Web or “The Web” for short, is even more recent. It is a collection of interconnected documents, which can be accessed via the Internet, initially invented by Tim Berners-Lee in 1990. The television concept on the other hand, goes back much further, with the first all-electronic television demonstrated in 1934. However, while the supporting technologies steadily evolved in the following 70 years, the concept itself did not change much conceptually. It always remained a platform on which you could receive a limited number of channels being broadcast. In the seventies, a first attempt towards television interactivity was taken with the introduction of Teletext, a textual layer showing a number of pages containing news, weather forecasts, program information, etc. While teletext became very popular, 30 years later, it is still the exact same service as introduced back then. Even now, people often need to wait up to 30 seconds for one page to appear.

While people experienced technological leaps on the Internet in terms of interfaces, interactivity, hardware development and screen quality, it became clear that the television platform was somewhat lagging behind. The Standard Definition (SD) television, with a resolution of 640x480 and a ratio of 4:3, started to look dated. However, already for years several groups around the world had been working on possible next generation standards with higher resolution outputs. Different formats, which could be called the precursors of High Definition (HD), where proposed by different consortiums. Unfortunately, there were some large issues preventing the adoption of the HD systems. Firstly, there was a tremendous lack of standardization and various groups just developed their own standards individually. Secondly, the HD format easily required over four...
CHAPTER 2. THE TELEVISION DOMAIN

Figure 2.1: TV sales in the Netherlands (http://media.immovator.nl/)

times the bandwidth of a standard definition (SD) analogue broadcast. The first and for some
time the only commercial system was operated in Japan and known as “Hi-vision”, featuring a
5:3 aspect ratio screen with 1,125 interlaced lines (1,035 active lines) at the rate of 60 fields per
second [255]. Hi-vision was, because of this huge bandwidth burden, broadcast via a satellite
system. However, already during the second half of the 20th century researchers began to ex-
periment with the digitization of analogue information, which would lead to a huge improvement
of quality, error detection, efficiency and throughput. Through this technique, data could also
be compressed without any loss of information. For HD broadcast this meant that very efficient
compression algorithms could reduce the necessary bandwidth for a digital signal greatly, making
it again much more commercially employable.

HD broadcasts were demonstrated around the world since the early 1990s. However, the first
real regular HD broadcasts only started on January 1, 2004. The reason for this continuously
postponing of HD television was due to the close relation between various parties in the television
distribution chain. Why would television distributors buy and broadcast HD content if no-one can
watch this content at home? Why would content producers generate HD content if the distributers
would not buy it? And why would we buy an HD television when there is no HD content available?
Luckily, this negative spiral was broken when a few companies decided to invest in HD content in
the beginning of 2004. In Figure 2.1 we see that for the Dutch market, once some HD content was
broadcast in that year, people started buying HD televisions, albeit with an expected reservation.
However, from 2005 we see that sales for HD Ready (the HD standard with 720 lines) televisions
started to rise steadily. Moreover, with the later adoption of the Full HD televisions (the HD
standard with 1080 lines) in 2006, we see that people more and more collectively choose for a high
resolution screen.

Due to the strong increase of bandwidth requirements of HD content and therefore its only
availability via digital streams, people were encouraged in switching over from an analogue tele-
vision system to a digital version. By doing so, most of them can now select from a number of
available digital HD channels. In Figure 2.2 we see an overview of Dutch households which have a
television subscription, together with the rising percentage of those households which have made
the switch to a digital subscription. As shown, at the end of the fourth quarter of 2008, already
53 percent of the television watching households switched to a digital television platform.
2.2 Interactive television applications

A revolution in the digitalization of video and television has been predicted for some time now [95]. Although at some point in the nineties, people again started doubting whether or not the mass public was really enthusiastically looking forward to digital television. Many studies were performed, but they turned out to be very contradictory in terms of estimated revenues and market share throughout Europe [138]. About one decade later, we can now finally safely assume we are standing at the verge of the television revolution. However, currently, the so highly anticipated television interactivity is still very limited. The main cause for this backlog is that a television still remains a dumb device to show images, without any intelligence, connectivity or data awareness whatsoever. So, in order to bring an extra layer of interactivity, another additional device is necessary. The set-top box (STB), as it was conveniently called, is such a small computer which can alter or augment the video stream send to the television. On top of that, it usually comes with a separate remote control, such that a user can interact with the system. Initially, the set-top box was a small embedded system including a CPU and TV-tuner running a middleware application on top of which developers could invoke their specific software. Set-top boxes came in all shapes and sizes and were developed by various hardware manufactures. As the number of different digital set-top boxes, with different hardware types and specifications, rose, so did the need for a stable unified middleware stack. This layer would shield off all technical features and peculiarities introduced by the various set-top boxes, while providing an integrated development environment in which third party application providers could build their end-user products. Through this middleware, applications could manage the user interface, user input, data streams and downloads, independent of differences in the underlying hardware.

Over the years two main middleware competitors arose providing such development environments. The first was OpenTV named after the homonymous company\(^1\). OpenTV is a middleware platform which provides an Application Programming Interface (API) enabling third party applications to run on a box independently of the underlying hardware. Applications running on OpenTV had to be written in C extended with proprietary OpenTV libraries. Because of this, OpenTV applications are able to run efficiently on lightweight hardware platforms which is a tremendous advantage for the service provider. According to OpenTV, currently, their middleware runs on more than 121 million devices worldwide\(^2\).

\(^1\)http://www.opentv.com/
Next to OpenTV, the Digital Video Broadcasting Project (DVB), a large consortium of over 270 broadcasters, manufacturers, network operators, etc., developed the Multimedia Home Platform or MHP. MHP has the identical purpose as OpenTV, with this difference that it is a Java based middleware specification. MHP is deployed in various countries compatible with all existing DVB standards. According to their Web site, currently 10 million receivers are MHP compatible. One of the most popular DVB specifications is DVB-J, which enables applications to behave similarly like a Java Applet. These STB Applets are also known as Xlets.

Companies developing STB applications can choose to develop their application for both OpenTV and MHP, and by doing so cover 99% of the set-top box market. In Figures 2.3 and 2.4
we see some examples of iTV (interactive TV) applications developed by Stoneroos Interactive Television\(^3\). In Figure 2.3 we see an example of an OpenTV application built to give extra reading in a classical music program. The STB provides an interface (which could be shown on demand) to get extra information about the operas, musicians, etc. In Figure 2.4 we see an interface used to quickly see which other related programs are currently playing on other television channels. Pressing one of these programs would immediately take you to that particular channel.

### 2.2.1 Electronic Program Guide

*The* application which is most popular and indispensable on any interactive television platform is the *Electronic Program Guide* or EPG. An EPG provides a digital version of the timetables indicating which TV and/or radio program starts where, and helps the viewer to find a program to watch or listen to. Traditionally, these timetables were and still are printed in news papers and magazines. However, with the introduction of more and more television channels, printed versions are usually restricting themselves nowadays to the standard set of legacy channels.

![Figure 2.5: Traditional EPG grid view (http://www.team-mediaportal.com/)](http://www.team-mediaportal.com/)

The first EPGs were and still are available through teletext. Most channels broadcasters follow the standard to reserve page ‘200’ of teletext as the place where they can put the overview of programs to come, as well as one or two days in the future. However, the EPG was also one of the first interactive applications on a STB, which introduced the EPG information in a *grid* view like shown in Figure 2.5. The grid view, but also more exotic variants of EPGs, show a list or group of channels together with their programs shown in a table providing an excellent overview of when programs start, end and how much they overlap. The most common form of EPG shows the channels, together with their programs, horizontally. Here a vertical ruler usually indicates the current time to give an idea how long a program is already playing. Examples of Dutch online EPGs of this type are “NU TV gids”\(^4\) and “VPRO gids”\(^5\). However, other EPGs show their channels vertically, as seen in the “RTL TV gids”\(^6\), “TV Gids.nl”\(^7\) and “Omroep NL gids”\(^8\).

---

\(^3\)http://www.stoneroos.nl/
\(^4\)http://www.nu.nl/tvgids/
\(^5\)http://epg.vpro.nl/index.php/eigengids
\(^6\)http://www.rtl.nl/service/gids/
\(^7\)http://www.tvgids.nl/nustraks/
\(^8\)http://gids.omroep.nl/home/
With the current and future increase in the number of channels (most STB systems already contain hundreds of TV programs simultaneously broadcast at hundreds of channels at any time of day), more advanced versions of EPGs are being devised. After all, as the number of channels keeps on growing, even EPGs will not make it easier to spot the programs you are interested in. E.g. the EPG shown in Figure 2.5 shows ten channels per page. Having for example 1,000 channels (which is not uncommon nowadays), already results in 100 of such pages. The next step in the evolution is therefore the so-called Personalized EPG or PEPG which tries to help the user in finding the programs and channels he or she will be most interested in [188]. Moreover, the newest EPGs already allow the user to for example select or group channels themselves, color favorite programs or even to personalize the EPG’s look and feel. For more information about the evolution of EPGs and PEPGs we would like to refer to [223].

2.3 Set-top box hardware

Preferably, STB systems running these interactive television applications have to be small and inexpensive. After all, people are not willing to pay a lot of money for a system which still has to prove its value. Therefore, to spark the initial adoption, set-top boxes were usually distributed for a small price together with a television subscription. As a result, an often reoccurring problem was the limited performance of these devices, severely restraining the type and capabilities of the applications. Often, developers needed to find creative solutions to get things running faster or more responsively. With limited STB resources, developers always preferred building their applications on top of the OpenTV middleware over MHP. The reason for this is simple. OpenTV uses the C programming language, which enabled them to squeeze more out of the hardware than was possible with the Java implementation of MHP (hence the difference in market share). Next to performance issues, other STB drawbacks included the absence of a data return channel (the ability to send data back to a server) disallowing 2-way communication, the lack of a local means of data storage, no separate data channel, etc.

Without a separate data line, transmitting data to the set-top box is only possible by pushing data embedded in the video stream. To do so, different specifications approaches exist, however one of the most known is the DVB specification making use of MPEG-2 data streams. Internationally, but dominantly in Europe, DVB (Digital Video Broadcasting) is the collection of internationally accepted open standards for digital television [83, 253]. DVB standards are maintained by the DVB Project, an international industry consortium with more than 270 members. Among these standards we find for example the protocols for the reception of a DVB signal e.g. DVB-C defines the cable protocol, DVB-T and DVB-T2 the terrestrial protocol, DVB-S the satellite protocol, etc. For each of these standards the physical layer and data link layer of the distribution system are defined. In DVB, data is transmitted in MPEG-2 streams with some additional constraints (DVB-MPEG), and/or a standard for temporally-compressed distribution to mobile devices (DVB-H).

The MPEG-2 transport stream, used by DVB, is a communications protocol for audio, video, and data [174]. Such a stream consists of one or more Packetised Elementary Streams (PES), which have a common time base. Every PES has a specific bandwidth and payload defined and is constituted out of packets of 188 bytes long. One program stream is constructed out of multiple PES streams, where for example one contains the video, one the audio and more can be assigned. Afterwards, a specific table tells which of these PES streams together define the program, and because of the common time base in every PES, these streams can be synchronized flawlessly. One or more of these PES streams can therefore also be used to send data along. The format of this data could be defined customary, as long as the receiving box could understand the format, which could differ depending on the middleware stack available on that box. OpenTV for example expected a specific data format in which there was not too much room for manoeuvring or stepping outside of the proposed stream model. If a data stream is available for a specific program, accessing data would work as follows: if the user presses a button during a specific scene, the stream can show additional information about current features of the program. This data can contain e.g. text or pictures fitting that particular program scene or a simple user interface. Through this layer also
very limited games like e.g. tic-tac-toe are possible to implement. Functionality-wise, MHP is not
different from OpenTV. For the development of MHP applications, the Java classes would be
send along with the DVB streams such that they could be executed by the Java Virtual Machine
(JVM) on the box. Data access is from then on very similar to the previously described scenario.

Many of these restrictions contributed to the initially very basic nature of the first interactive
tv applications. However, with current advances in hardware and ever smaller integrated circuits,
stbs are getting smaller, usually include a return channel and become more powerful by the day,
allowing more extensive and varied applications. Some of the newest generation of televisions even
include stb hardware within the television itself, making the extra device superfluous.

2.4 Television 2.0

Clearly, there are many parallels which can be drawn between the evolution of the Internet on
the computer platform and the sprouting interactivity on the television/set-top box combination.
Currently, the main culprit holding back the emerging revolution of the television in comparison
to the computer platform, is the absence of decent connectivity to the rest of the world. Just
like the evolution of hard- and software on the personal computer, which really skyrocketed when
computers got connected to the Internet, a connected tv could suddenly open a new dimension
to the information highway. It can for example allow for known functionality and features like
blogging, surfing, emailing, chatting, etc. with which we already feel so familiar. However, also
many new ideas and features can be thought of, specifically exploiting the unique character of the
tv platform. In this respect, the television system would undergo the same evolution as
the Internet, albeit with a tremendous kick start.

A first important parallel with the evolution of the Internet, involves the general perception
of television content. Currently, people are very used to having 20 or 30 channels, and each
of these channels broadcast programs at fixed times. However, once connected to the world,
we would see a radical change in television content provisioning. Suddenly, programs would for
example be available on demand. For a small price, you can select whatever you want to see
at the time and place that fits you best. This basically boils down to a shift from a broadcast-
centered platform towards a user-centered environment where people can decide for themselves
what to watch. Broadcast channels will still exist, however they will get though competition
from thousands of potential Web channels. Just like we experienced with the radio medium,
where hobbyists and professionals suddenly were setting up their own online radio stations and
shoutcasts, others will start their own television Web channel with visual content. Online we
can already see this trend on thousands of initiatives like e.g. channelchooser.com\(^9\) with 3400+
channels, wwtv.com\(^10\) with 3000+ channels, tvweb360.com\(^11\) with 1400+ channels, etc. We can
safely expect that much more of such channels will sprout if suddenly all of them can directly be
brought to the living room. Next to Video-On-Demand (VOD) and various Web channels, a third
source of data will manifest itself, namely user generated content. On the Internet we witnessed the
substantial social evolution introduced by the Web 2.0 initiative [187], which eventuated in people
sharing pictures on Flickr, movies on YouTube, tweets (short -what are you doing- messages) on
Twitter, bookmarks on del.icio.us, etc. Similarly, a television platform could be an entry point
to share videos and pictures with a community. After all, a lot of people still prefer connecting
cameras directly to the television as it usually feels more intuitive than a computer, where still
some basic operating system knowledge is required. The television lends itself perfectly to connect
your camera, review your content, make a selection and send it to some friends or a predefined
sharing group. With respect to community functionality, the television system could potentially
unlock numerous new features, besides sharing user generated content, like recommending a piece
of content or a channel to others, watching a program together as a group while every participant
resides at home whilst communicating via voice-over-ip, etc.

\(^9\)http://www.channelchooser.com/
\(^10\)http://wwtv.com/
\(^11\)http://tvweb360.com/
CHAPTER 2. THE TELEVISION DOMAIN

2.4.1 Television user behavior

Clearly, the previous section showed that a smart television platform can create a whole new set of possibilities for the end-user. Moreover, the television medium connected to the world wide Web would unleash a tremendous revolution in our daily TV watching behavior. However, there are also some sharp discrepancies in comparison to how the Internet evolved. The most important difference manifests itself with the very different behavior that people exhibit, or the expectancies they have when watching television in comparison to the average computer utilization. These differences can be elucidated by various social behavior. The most notable is the discrepancy between active or *leaning-forward* and passive or *leaning-backward* behavior. People turn to their computers when they actively want to search for information or utilize services, setting aside some exceptions. Still, this act of information gathering requires people to actively search, click and browse until they are satisfied with the results obtained. Watching television on the other hand is more considered to be an act of pursued leisure, again setting some exceptions aside. Moreover, in various cases the attention to the television set is minimal. E.g. people often fall asleep while watching television, some have the habit to turn it on while doing the daily housekeeping or some even turn it on in the background just to get more liveness in the house, clearly indicating a more passive use.

The way people approach a TV differs a lot from the intention they have when turning on their PC. In the past, there were a lot of studies looking at patterns and influences on television viewing behavior. In [156] for example, Lull made an ethnographic overview of the uses of television. In [150], Kubey et al. present various empirical studies showing relations between people and the television platform, underpinned with surprising statistics like e.g. the average number of years a person spends in his life on watching television. In [8], Anderson et al. tried to determine some long term relations between preschool television viewing and adolescent achievements and behavior. Conclusions were formulated like: “watching informative television when you are young often leads to higher participation in academic activities”. In [259], Wonneberger et al. exposed various patterns of viewing behavior and look at some models to explain them. Some of these studies point to another discrepancy between the TV set and a regular computer, namely the aspect of group behavior [250] and [172]. After all, television viewing often occurs in families or other social groups. This behavior differs a lot from general computer utilization which is usually a more personal activity.

2.4.2 Looking at the future

Due to these differences in user behavior, we are convinced that some approaches for applications which worked on the Internet might not apply equally well on the TV platform. We believe that an application running on a television/STB combination for example must be much more context-aware to serve the user well in different circumstances. However, we expect the true power to lie in the combination of platforms, yet in an interoperable way. In such a case, it is the user who remains in control over what he wants and how he wants it. Therefore, we advocate an integrated approach serving the user depending on his current constraints in terms of available devices and/or willingness to interact. In Figure 2.6 we see a conceptual overview of various spaces, which we identified as the most probable interactivity entry-points and their interrelationships. In the virtual *Web space* users are actively interacting with content and applications by searching, browsing and performing numerous information intensive tasks. In the *Mobile space*, sessions are shorter in time, where users aim at staying up-to-date and consume small items from e.g. RSS feeds of news headlines and weather forecasts. It can also be used to quickly select, control or augment interesting items that can later be consumed in full in any of the other spaces. After all, devices used here are limited in terms of bandwidth and screen size. Typically, in the *physical space*, the focus is on consumption of media and passive entertainment, where a push strategy for content is the typical pattern of user interaction with the devices. In order to provide a personalized experience to the user in all three spaces we argue that we should not aim at improving the simulation of each of the three spaces in the other two, but rather to integrate and use the tools,
2.4. TELEVISION 2.0

Figure 2.6: Different spaces for different needs

devices and the interaction abilities in a complementary way. In other words, our vision is that the Web environment could be used as a remote control to the physical space, where the mobile devices can be used as carriers of the information, identity and services between the virtual and physical spaces. For example, the user can search and browse for content on the Web (space), set up preferences, define usage contexts and identities via his or her mobile, while benefitting from both at the physical consumption space. Both the Web environment as the mobile device could also be used to remotely control devices at home (e.g. set a program to be recorded on your Personal Video Recorder (PVR) while being at work or on the train). Previously, Venkatesh et al. configured the future networked home similarly in terms of a “living space”. This living space then further consists of three structural components: the social space, the physical space and the technological space [245]. The social space represents among others all activities, actions and emotions of the members of the household. It for example includes the time spent on those activities, interactions between the members of the family, etc. The physical space is the most stable space and refers to the physical location in the home including for example the kitchen, bedrooms, bathrooms, etc. The physical space here reflects the same physical space we presented in Figure 2.6. The technological space on the other hand consists of the household technologies that are embedded in the physical space and used by the members of the family as part of the social space [245]. Conceptually, this technological space represents the combination of both our virtual and mobile space.

Further, in Figure 2.7 we see an overview of the current and expected steps in the evolution of television capabilities in terms of interactivity. Starting at the bottom, after the general Analogue Television systems, the first digital systems appeared which were able to show some theme channels and a first crude EPG. However, this Digital Television systems did not have a return channel yet. Later, we found the Enhanced Television sets which did have a return channel showing some first forms of service-based interactivity allowing for example to buy a video or program. This is more or less where the mainstream market resides right now\(^\text{12}\). In the near future we will slowly

\(^{12}\text{Now, at the end of 2009}\)
convert to the Interactive Television which will allow more elaborate interactivity, enable direct communication with peers and allowing the formation of communities. However, our real vision is a Personalized Television system which allows for personalized content depending on the current user and context, while available on different integrated devices.

In [37], Bernhaupt et al. look at the future of interactive television in the living room, from the Human Computer Interaction viewpoint. Further, Bernhaupt et al. present the findings of two ethnographic studies embedded into two broader projects on interactive television in the home environment. In this context, especially the second study was interesting as they investigated 1) the concept of an extended home (can smart and connected devices provide the background for ubiquitous connectivity and communication?); 2) the shared experience (which activities/technologies/content do people already share at home?); and 3) new kinds of interaction techniques for the living room (which innovative technologies can make life easier in the future?). From the different observations made, they found among others that “Security and personalization were seen as important aspects in the development of new kinds of interactive services for the home”. Further, Bernhaupt et al. state that ongoing convergence of content from personal computers, mobile phones, and television opens more possibilities for bringing personalization to television in the home context.

### 2.5 Modeling the domain

Looking at the success of technologies and platforms, people want the TV set to evolve in terms of data provisioning and interactivity. However, we assume that most people at the same time do not want the TV to transform into a second personal computer adding complexity to a simple process like “turn on the TV and start enjoying”. It requires the best of both worlds which will lead to the next step in home entertainment systems: A state-of-the-art information system connected to the Web and able to present and retrieve content from various online data sources, while presenting it in such a way that a regular user can still enjoy his television evening in a personalized but unobtrusive way. However, with rising complexity of such a system, the need for an exhaustive model of all actors (e.g. content, users, devices, events, etc.) rises as well. We need to know which user uttered which request on which device, before we can decide which content qualifies best to be returned. Therefore, only with a well-structured model and clear semantics of these actors, a system could be able to, given circumstances and limitations, exhibit the best attainable behavior.
2.5. MODELING THE DOMAIN

2.5.1 Metadata: a definition

When on one hand the amount of data grows to enormous sizes, and on the other hand the requirements on interactivity, navigation and filtering rise, the need for well-structured item descriptions becomes undisputable. These “item descriptions” are pieces of structured information which give us facts and statements about that item which were not included within the item itself. The more scientific term for such descriptions is metadata. Defining a real definition however is hard. Wikipedia describes it as “data about other data”, while the World Wide Web Consortium (W3C) and more specifically Tim Berners-Lee describes it as “Machine understandable information about Web resources or other things” [34].

Having items, in this domain television programs and other TV related objects, which are nicely described by means of various metadata properties, helps us to understand what these items are exactly about. Such metadata can tell us for example the title of the program, a larger explanation or synopsis, which are the participants (like e.g. actors, presenters, directors, guests, etc.), the program’s genre, duration, video and audio quality, etc. However, considering that this metadata has to be created by an editorial staff, sent to content providers, being interpreted by software like for example an EPG, etc., some foreknowledge of the structure is desired. Moreover, this structure should be understandable for both people and machines as both will have to deal with this information. In order for metadata descriptions to be machine readable they must therefore adhere to some form of agreement, a so-called “description schema” or “metadata schema”. Such a schema defines the structure of the metadata by identifying the allowed types, fields and relations which are necessary to describe a specific item in a particular domain. The idea of such schemas originally comes from the database field where a schema describes the layout of the tables.

2.5.2 State-of-the-art: Audiovisual Description schemas

In complex audiovisual content management and delivery chains (like the TV system), metadata undoubtedly plays an important role. With in the future potentially millions of content providers, broadcasters, service providers, end-users, devices and formats, everything stands or falls with the quality of the descriptions accompanying the data being send back and forward between all these different process actors. The content descriptions are the oil in the engine starting from content creation until content consumption. To clarify with some simple examples: a content provider must have the means of making valid descriptions such that the subsequent parties in the content chain understand what they are dealing with. A service provider needs to have exact descriptions in terms of size and quality of the content to estimate the load on the networks. The STB should be able to interpret the descriptions send along, such that the end-user can search for sport programs on Saturday. To accomplish such functionalities, we need well-structured metadata providing all the information and links which are there to know. The reason for this information to be well-structured, is because a computer program should be able to understand the structure and the semantics of this descriptions, such that it can meet the various requirements in each subsequent step of data processing.

With the mass creation of different types of media like texts, pictures, videos, music, etc. and the increasing demand for intelligent navigation and filtering, the need for such well-thought through content descriptions was never higher. Therefore, throughout the years, research in specific domains has led to the creation of various content description schemas. However, their purpose can vary reasonably depending on the place of deployment within the content chain. Some descriptions like TV-Anytime [238] and XML-TV [260] focus primarily on business-to-consumer (B2C) applications, while descriptions like P/Meta [88] focus more on business-to-business (B2B) employments. However, the large diversity in different description schemes has led to the situation where choosing the right schema for a specific task has become increasingly difficult. Moreover, the overlap which sometimes exists between these schemes, further aggravates the choice of the best schema for the application. In the next section we give an overview of the most well-known description schemes for audiovisual content and summarize their goals, advantages and disadvantages with respect to our endeavor to create flexible and well-structured TV program descriptions.
MPEG-7

MPEG-7 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group) [177], the committee that also developed the successful standards known as MPEG-1 (1992), MPEG-2 (1994) and the MPEG-4 standard (Version 1 in 1998, and version 2 in 1999). Where the MPEG-1 and MPEG-2 standards have enabled the production of widely adopted Video CD, MP3, Digital Audio Broadcasting (DAB), DVD, digital television (DVB and ATSC), and many Video-On-Demand trials and commercial services, MPEG-4 is the first real multimedia representation standard, allowing interactivity and a combination of natural and synthetic material coded in the form of objects (it models audiovisual data as a composition of these objects).

The MPEG-7 standard [176], formally named “Multimedia Content Description Interface”, provides a rich set of standardized tools to describe multimedia content. Its main information retrieval objective is to quickly and efficiently search for various types of multimedia documents of interest to the user. Further, it is fit to filter in a stream of audiovisual content descriptions to receive only those multimedia data items which satisfy the user preferences. E.g. a code in a television program triggers a suitably programmed PVR (Personal Video Recorder) to record that program.

To meet the standard’s objections, a set of different tools were conceived [205]. With these tools, a MPEG-7 description creator can build valid MPEG-7 descriptions consisting of one or more instantiated description schemes, which together describe the data at hand. These tools are classified like:

- Descriptors (D): A descriptor is a representation of a feature, where a feature is a distinctive characteristic of the data. Hence, the descriptor defines the syntax and the semantics of the feature representation. E.g. a time code for representing duration, a character string for representing a title, etc.

- Description Schemes (DS): A description scheme specifies the structure and semantics of the relationships between its components, which may be both descriptors and description schemes. E.g. a movie still structured as separate scenes and shots including various descriptors at scene and shot level.

- Description Definition Language (DDL): DDL is a language that allows the creation, extension and modification of description schemes and possibly descriptors.

- System Tools: Tools related to the binarization, synchronization, transport and storage of descriptions, as well as management and protection of intellectual property.

Further, MPEG-7 consists out of eight main parts [205]:

- MPEG-7 Systems: Specifies the tools that are needed to prepare MPEG-7 descriptions for efficient transport and storage (binarization), to allow synchronization between content and descriptions and the tools related to managing and protecting intellectual property.

- MPEG-7 Description Definition Language (DDL): Specifies the language for defining new descriptors (D) and description schemes (DS). The DDL is based on XML Schema Language. But because XML Schema Language has not been designed specifically for audiovisual content description, there are certain MPEG-7 extensions which have been added. As a consequence, the DDL can be broken down into the following logical normative components:
  - The XML Schema structural language components.
  - The XML Schema data type language components.
  - The MPEG-7 specific extensions.

- MPEG-7 Visual: Specifies the descriptors and description schemes dealing exclusively with visual information that covers following basic features: color, texture, shape, motion, localization, and face recognition. Each category consists of elementary and sophisticated descriptors.
2.5. MODELING THE DOMAIN

- MPEG-7 Audio: Specifies the descriptors and description schemes dealing exclusively with audio information. Utilizing those structures are a set of low-level Descriptors, for audio features that cut across many applications (e.g. spectral, parametric, and temporal features of a signal), and high-level Description Tools that are more specific to a set of applications. Those high-level tools include general sound recognition and indexing Description Tools, instrumental timbre Description Tools, spoken content Description Tools, an audio signature Description Scheme, and melodic Description Tools to facilitate query-by-humming.

- MPEG-7 Multimedia Description Schemes (MDS): Comprises the set of Description Tools (Descriptors and Description Schemes) dealing with generic as well as multimedia entities. Generic entities are features, which are used in audio and visual descriptions, and therefore ‘generic’ to all media. These are, for instance, ‘vector’, ‘time’, textual description tools, controlled vocabularies, etc. Apart from this set of generic Description Tools, more complex Description Tools are standardized. They are used whenever more than one medium needs to be described (e.g. audio and video.) These Description Tools can be grouped into five different classes according to their functionality:
  - Content description: Representation of perceivable information.
  - Content management: Information about the media features, the creation and the usage of the AV content.
  - Content organization: Organization and modeling of multimedia content collections.
  - Navigation and access: Specification of summaries and variations of the AV content.
  - User interaction: Description of user preferences and usage history pertaining to the consumption of the multimedia material.

Furthermore, the MDS tools provide constructs for Classification Schemes (CSs) and controlled term hierarchies. The classification schemes provide a language independent set of terms that form a vocabulary for a particular application or domain. Controlled terms are used in descriptions to make reference to the entries in the classification schemes. Allowing controlled terms to be described by classification schemes offers advantages over the standardization of fixed vocabularies for different applications and domains, since it is likely that the vocabularies for multimedia applications will evolve over time.

- MPEG-7 Reference Software: Includes software corresponding to the tools included in the standard. The eXperimentation Model (XM) software is the simulation platform for the MPEG-7 Descriptors (Ds), Description Schemes (DSs), Coding Schemes (CSs), and Description Definition Language (DDL). Besides the normative components, the simulation platform needs also some non-normative components, essentially to execute some procedural code to be executed on the data structures. The data structures and the procedural code together form the applications. The XM applications are divided in two types: the server (extraction) applications and the client (search, filtering and/or transcoding) applications.

- MPEG-7 Conformance Testing: Defines guidelines and procedures for testing conformance of MPEG-7 descriptions and terminals.

- MPEG-7 Extraction and Use: Provides information on the extraction and use of some description tools, notably giving insight into the Reference Software. This part is a technical report and not a standard. It basically corresponds to the textual version of the visual part of the XM, which describes all the normative and non-normative visual tools implemented in the XM software.

Audiovisual data content that has MPEG-7 descriptions associated with it, may include: still pictures, graphics, 3D models, audio, speech, video, and composition information about how these elements are combined in a multimedia presentation (scenarios). A special case of these general data types is facial characteristics. The MPEG-7 descriptions do, however, not depend on the
CHAPTER 2. THE TELEVISION DOMAIN

Ways the described content is coded or stored. It is possible to create an MPEG-7 description of
an analogue movie or of a picture that is printed on paper, in the same way as of digitized content.

MPEG-7 allows different granularity in its descriptions, offering the possibility to have different
levels of discrimination. Even though the MPEG-7 description does not depend on the (coded)
representation of the material, MPEG-7 can exploit the advantages provided by MPEG-4 coded
content. If the material is encoded using MPEG-4, which provides the means to encode audiovisual
material as objects having certain relations in time (synchronization) and space (on the screen for
video, or in the room for audio), it will be possible to attach descriptions to elements (objects)
within the scene, such as audio and visual objects.

```
type=complete>
  <ContentDescription xsi:type=ContentEntityType>
    <MultimediaContent xsi:type=ImageType>
      <Image>
        <MediaLocator>
          <MediaUri>http://w3.tue.nl/uploads/RTEmagicC_logo.gif.gif
        </MediaUri>
      </MediaLocator>
      <CreationInformation>
        <Creation>
          <Title xml:lang=en>The TU/e Logo</Title>
          <Creator>
            <Role href=urn:mpeg:mpeg7:cs:RoleCS:AUTHOR>
              <Name xml:lang=en>Author</Name>
            </Role>
            <Agent xsi:type=OrganizationType>
              <Name>Eindhoven, University of Technology</Name>
            </Agent>
          </Creator>
          <RelatedMaterial>
            <MediaLocator>
              <MediaUri>http://w3.tue.nl/nl/</MediaUri>
            </MediaLocator>
          </RelatedMaterial>
        </Creation>
      </CreationInformation>
    </Image>
  </MultimediaContent>
</ContentDescription>
</Mpeg7>
```

Figure 2.8: Example of MPEG-7 Metadata description

Because the descriptive features must be meaningful in the context of the application, they
will be different for different user domains and different applications. This implies that the same
material can be described using different types of features, tuned to the area of application. To
take the example of visual material: a lower abstraction level would be a description of e.g. shape,
size, texture, color, movement (trajectory) and position (where in the scene can the object be
found?); and for audio: key, mood, tempo, tempo changes, position in sound space. The highest
level would give semantic information: “This is a scene with a barking brown dog on the left
and a blue ball that falls down on the right, with the sound of passing cars in the background”.
Intermediate levels of abstraction may also exist.
2.5. MODELING THE DOMAIN

MPEG-7 addresses many different applications in many different environments, which means that it needs to provide a flexible and extensible framework for describing audiovisual data. Therefore, MPEG-7 does not define a monolithic system for content description but rather a set of methods and tools for the different viewpoints of the description of audiovisual content. Having this in mind, MPEG-7 is designed to take into account all the viewpoints under consideration by other leading standards such as, among others, TV-Anytime [238], Dublin Core [82], SMPTE Metadata Dictionary [195], and EBU P/Meta [88]. These standardization activities are focused to more specific applications or application domains, whilst MPEG-7 has been developed as generic as possible. MPEG-7 also uses XML as the language of choice for the textual representation of content description, as XML Schema has been the base for the DDL (Description Definition Language) that is used for the syntactic definition of MPEG-7 Description Tools and for allowing extensibility of Description Tools. Considering the popularity of XML, usage of it will facilitate interoperability with other metadata standards in the future [176]. In Figure 2.8 we see a short example of an MPEG-7 description of a logo which was made by (author) our university.

Escort 2.4

In the mid-nineties, the European Broadcasting Union (EBU) designed Escort 2.4 to respond to the broadcasters’ need to have more reliable, easily accessible and internationally comparable data in many areas of their activities [86]. By means of a number of elementary dimensions, Escort 2.4 classifies and annotates Radio and Television (RTV) programs. These elementary dimensions can be divided in two groups: program conceptual dimensions and administrative productional dimensions. The first group concerns data relevant for the description of the form and content of the program, and consists of:

- Intention: Intention of the RTV program, e.g. information, entertainment, etc.
- Format: Format of the program, e.g. bulletin, magazine, hosted show, etc.
- Content: Content of the program, e.g. news, sports, cookery, etc.
- Target group: The group targeted with this program e.g. young children, immigrant groups, farmers, etc.
- Origination: Where the program originated from, e.g. TV studio, cinema industry, theater, etc.
- Language: The languages used in the program
- Participation: A dimension reserved for the purpose of classifying data regarding the participation of representatives of certain relevant (pressure) groups e.g. political parties, linguistic minorities, regions etc.

The second group of dimensions concerns data of a more administrative nature and refers to the way in which the program was produced/acquired, how it was scheduled and transmitted, how it was received by the audience and what costs were involved. The central dimensions are:

- Administrative data: E.g. the unique id of the production, etc.
- Program Acquisition data: E.g. output mode, department, originating country, etc.
- Scheduling data: E.g. duration, frequency, transmission cycle, etc.
- Transmission data: E.g. month of the year, day of the month, etc.
- Listening/Viewing data: E.g. audience behavior, appreciation, etc.
- Financial data: E.g. program costs, funding, rights, etc.
In each dimension, only one code can be given. The digits of any dimension code form a hierarchy. The first digit stands for the main class, the second refers to a sub-class of the first class, the third is a sub-class of the second, etc. If the specification does not fit the data of the program dimension concerned, the general term ‘Other’ can be used. Normally the dominant characteristic determines the classification to be used. If no single quality prevails, the general term ‘Mixed’ can be applied. If the data concerned are not known, the general term ‘Unknown’ can be used. Information about content and broadcasting operations is generated and processed as defined in a series of technical specifications from the EBU and other standards organizations such as the European Telecommunications Standards Institute (ETSI) and the Society of Motion Picture and Television Engineers (SMPTE).

Escort 2.4 initially represented only a small subset of the overall information managed within broadcasting facilities. Since production and scheduling systems nowadays handle more data than was originally defined in Escort 2.4, they started the development of Escort 2007 [85]. The intention of Escort 2007 is to remain a reference for the exchange of comparable data in the field of broadcasting statistics and audience research. Escort 2007 may also be used for finance and accounting, marketing and compliance reporting. With the evolution of technology, it is expected that the deployment of future personal digital video recorders will change the way television is consumed, which in turn is likely to have an impact on scheduling and advertising strategies. Therefore, the new standard has been prepared to deal with potential future extensions.

TV-Anytime
The global TV-Anytime Forum [238] is an association (of more than 100 member organizations and 500 individuals) which seeks to develop specifications to enable audiovisual and other services based on mass-market high volume digital storage in consumer platforms. The TV-Anytime Forum was formed in September 1999 and started working to develop open specifications designed to allow consumer electronics manufacturers, content creators, telecommunication companies, broadcasters and service providers to exploit local storage.

TV-Anytime allows the consumer to find, navigate and manage content from a variety of internal and external sources including, for example, enhanced broadcast, interactive TV, Internet and local storage [92]. It defines a standard way to describe consumer profiles including search preferences to facilitate automatic filtering and acquisition of content by agents on behalf of the consumer. The need to associate metadata with content to facilitate human and automated searching for content of interest, is great. Such metadata includes descriptive elements and attractors to aid the search process as well as elements essential to the acquisition, capture and presentation processes, content rights, formats, duration, etc. Many of these descriptive elements can be found in electronic program guides and Web pages. The process of creation and evolution of metadata for an individual content item may involve many organizations during the course of creation, distribution and delivery to the consumer. Thus, TV-Anytime endeavors to define a common metadata framework and a standard set of metadata elements in order to ensure a high level of interoperability within the chain from content creation to content delivery.

Figure 2.9 shows the flow of metadata and content through various stages of creation and delivery to the end consumer. This model clearly identifies the separation of the processing of metadata and content while at the same time illustrating the parallels between the processing of metadata and content. User profile and history metadata is generated during the selection and presentation process. The content creation process represents the production of a piece of content or a program. During the production process, the program content is created and information about the program may also be captured. At this stage, however, the metadata is unlikely to be in a form that can be directly exposed to a user, i.e. some form of editing will be required before the description of the program can be published. Once content has been created, the content is then available for publication by a content publisher. This could be, for example, as part of a broadcast service or as a publication on the Internet. The content publishing process defines instantiations of programs. In other words, one output from the content publishing process is information about where the program can be found. In the broadcast case, this means a schedule for the services
that are published. The content selection and presentation process may occur through the direct
involvement of the consumer or may be performed on the consumer’s behalf by a software agent.
In order for a software agent to function correctly, metadata describing the consumer and his
preferences will need to be provided to the content selection process. This may be either inferred
from the consumer’s past history of content selection or by the explicit specification of preferences
by the user (or a combination of the two).

Figure 2.9: TV-Anytime Metadata and content flow (Taken from ETSI TS 102 822-3-1)

The cornerstone of TV-Anytime metadata is the Content Reference IDentifier (CRID), de-
scribed in ETSI TS 102 822-4 dealing with content referencing. As a content reference identifier,
the CRID refers to a piece of content, though in some cases it may refer to one or more other
CRIDs. TV-Anytime metadata consists of two phases. Phase I deals with program metadata
consisting of four main categories (content description metadata, instance description metadata,
consumer metadata and segmentation metadata). Phase II takes care of relations between different
resources. In both phases all relations and references are facilitated by CRIDs.

For the purpose of interoperability, the TV-Anytime Forum has adopted XML as the common
representation format for metadata. XML offers many advantages: it allows for extensibility,
supports the separation of data from the application and is widely used. XML schema is mainly
used to represent the data model and the encoding of the metadata. Also, the fact that XML is
so widely adopted enables a great support in terms of available tools and knowhow. Furthermore,
TV-Anytime makes extensive use of the MPEG-7 constructs. It uses the MPEG-7 Description
Definition Language (DDL) to describe metadata structure as well as several MPEG-7 data types
and various Classification Schemes.

As previously mentioned, the Classification Scheme (CS) is an MPEG-7 tool for the provision
of controlled terminology for use in classification. The MDS specification shows how Uniform
Resource Names (URNs) can be used to uniquely identify CSs and terms within CSs, as well as the
use of CS aliasing to provide a more concise, application-specific way of referring to classification terms. Just like the classifications in MPEG-7, an informative set of Classification Schemes has been developed by TV-Anytime as well to provide a universally applicable default set of classification terms. The TV-Anytime classification hierarchies are based on the original hierarchies constructed in Escort 2.4 [86]. In addition to these default CSs, other CSs may be created and used to meet specific regional or other special requirements. These default CSs are presented as well-formed XML instance documents complying with the ClassificationScheme fragment defined in ETSI TS 102 822-3-2.

XML-TV

The XML-TV Project [260] covers a set of (mostly Perl) utilities to manage your TV viewing by retrieving and managing program metadata to be used in EPGs. Mainly, these utilities are able to retrieve and store TV listings in the XML-TV-Format, which is based on XML. The idea is to separate out the back-end (getting the listings) from the front-end (displaying them for the user), and to implement useful filtering operations when reading and writing resulting XML documents. At present, various different back-ends grab TV listings for countries like Australia, Belgium, Britain, France, Italy, Netherlands, North America, Spain, Sweden, etc. The scripts responsible for the retrieval of EPG data basically parse HTML Web pages to fetch the information from online program listings. While this might look like obtaining information illegitimately via scraping the screen, XML-TV’s policy incorporates a clause that forbids any further use of that particular content source if the source owner explicitly announces that they have a problem with these practises.

The XML format used by XML-TV differs from most other XML-based TV listings formats in that it is written from the user’s point of view, rather than from the broadcaster’s. It does not divide listings into channels, instead all the channels are mixed together into a single unified listing. Each program has details such as name, description, and credits stored as supplements, but metadata like broadcast details are stored as attributes. There is support for listings in multiple languages and each program can have ‘language’ and “original language” details. The Document Type Definition (DTD) of this format can be found in Appendix A.1.1.

IPTC ProgramGuideML

The International Press Telecommunications Council [136] (IPTC), is a consortium of the world’s major news agencies and news industry vendors. It develops and maintains technical standards for improved news exchange that are used by virtually every major news organization in the world. Currently about 70 companies and organizations from the news industry are members of the IPTC. Most of IPTC’s current work involves XML-based business-to-business standards for sharing news, and development of advanced metadata to describe and classify news text, photos, graphics, videos and other media. NewsML is one of the major standards developed by IPTC to provide a media-independent, structural enveloping framework for multimedia news. It was recently superseded by NewsML-G2, although IPTC plans to support NewsML indefinitely.

However, next to NewsML, the IPTC also developed the less known ProgramGuideML specification [193]. The main requirements of ProgramGuideML is to express Radio and TV listing in a media-independent way. ProgramGuideML is developed for news providers such as newspapers and news agencies to handle Radio and TV program information as a program unit or listing table. ProgramGuideML uses XML technology and composes radio and TV program information in a simple and powerful structure. It handles arbitrary mixtures of media types, formats and languages. ProgramGuideML documents can be wrapped in NewsML or distributed autonomously. Further, it adopts the controlled vocabulary mechanism of NewsML.

ProgramGuideML document is a XML document, which should be valid with respect to the ProgramGuideML DTD or ProgramGuideML Schema. ProgramGuideML documents can be built up from the contents of multiple physical files by using entity references as described in the XML specification of pointers within the ProgramGuideML document. ProgramGuideML can integrate
2.5. MODELING THE DOMAIN

optional medias such as text, video, audio, etc., and new upcoming media types in the future. Further, to express program information, ProgramGuideML makes use of TV-Anytime constructs by exploiting the TV-Anytime namespace. While TV-Anytime fits for the usage in Web and broadcast media, ProgramGuideML has its specified information to stipulate the layout when printing the program information and allows for the usage of TV-Anytime in that print media. Therefore, news agency could receive program data from broadcasters by TV-Anytime and deliver them to newspapers as ProgramGuideML, with attaching layout information to match each print media.

SMPTE Metadata Dictionary

The SMPTE Metadata Dictionary structure defined in the SMPTE-335M standard [195] covers the use of metadata for various types like e.g. video, audio, and data in their various forms. This standard is built by the SMPTE, the Society of Motion Picture and Television Engineers (SMPTE) [219]. SMPTE was founded in 1916 to advance theory and development in the motion imaging field and serves as the widely accredited and globally-respected industry standards-setting body. SMPTE’s Engineering Documents, including Standards, Recommended Practices and Engineering Guidelines, are prepared by SMPTE’s Technology Committees.

The metadata dictionary structure provides flexibility in capturing metadata and exchanging it among applications through a standardized hierarchy of universal labels for the metadata elements, grouped to aid their management within a small but comprehensive number of classes. Metadata classes are collections of metadata elements with common characteristics or attributes. The actual metadata information described by the metadata element is the metadata value. The dictionary also contains information on the required format of metadata values and the allowable range of values (if applicable) either as a list or as a bounded range.

Within the metadata dictionary, metadata elements are organized into a hierarchical structure, where each is assigned to a metadata class. The initial set of metadata classes in this standard consists of:

- Class 1: Identification and location
- Class 2: Administration
- Class 3: Interpretive
- Class 4: Parametric
- Class 5: Process
- Class 6: Relational
- Class 7: Spatio-temporal
- Class 13: Organizationally registered for public use
- Class 14: Organizationally registered as private
- Class 15: Experimental

The number of metadata classes can be extended in the future to a maximum of 127. Although dictionary classes can be populated with any metadata (such as that associated with still images, audio, graphics, etc.), additional new classes may be created up to that limit to deal with specific metadata characteristics or attributes. Each of these classes further consists of metadata that describes information about the type or metadata that is relevant to its application. Each metadata element is listed by name, with a definition of what it is, its data type, length, reference to existing standards, where appropriate, and a unique 8-byte key [220].
EBU P/Meta

The European Broadcasting Union (EBU) is the largest association of national broadcasters in the world [87]. EBU promotes cooperation between broadcasters and tries to facilitate the exchange of audiovisual content. EBU works to help develop many new radio and TV systems like e.g. Radio Data System (RDS), Digital Audio Broadcasting (DAB), Digital Video Broadcasting (DVB), High-Definition TV (HD-TV), etc. They promote open technical standards and interoperability for the benefit of broadcasters and consumers, and explore the opportunities presented by new technologies. Further, EBU is responsible for the organization of various large-scale events from Skiing World Championships to concerts such as the Proms, the Eurovision Song Contest, rock and pop festivals.

Furthermore, to facilitate the exchange of program related metadata and content, EBU launched the project group P/Meta, which has been working since 1999, to create a standard vocabulary for program exchange in the professional broadcasting industry [88]. P/Meta is a universal standard for metadata exchanges between professional media organizations. Various bodies across the industry have been engaged in parallel work, including SMPTE [219], MPEG [177], Dublin Core [82] and TV Anytime [238], but harmonization of vocabularies has proved elusive. The P/Meta scheme therefore tries to identify a common core which refers to many of these other initiatives [90], but recognizes that organizations have different internal needs, and that to standardize everything is unattainable and undesirable. P/META is therefore designed to be as flexible as possible in implementation while retaining consistency of meaning. It is technology-independent, and can be used in applications to create XML documents, embed metadata in file formats such as MXF [192], or simple Word templates. P/Meta does not support storage of information, it is not intended as a database scheme.

The P/META Scheme [89] is basically a set of definitions which provide a semantic framework for the information which is typically exchanged along with audiovisual material. It provides metadata to support the identification, description, discovery and use of essence in Business-to-Business (B2B) transactions. Further, it includes the identification of concepts (simple or complex) that are referenced by P/META names and P/META Identifiers. The P/META Scheme describes a collection of Attributes. It suggests Sets of Attributes for exchange. The P/META Scheme, currently, is comprised out of four main parts:

- Attribute Definition Table: The dictionary of agreed Attributes (currently 213 discrete Attributes) and their definitions required to support the exchange of content. Attribute examples are e.g. PROGRAMME\_TITLE for the program’s title, ROLE\_NAME for the role played by a contributor in a production, etc. See [88] for the complete list.

- Attribute Value Table: The specific values that may allowably be held by certain Attributes and the definition of those specific values. A value example is e.g. ‘Consultant’ as value for the attribute ROLE\_TYPE\_CODE.

- Set Definitions: The collection of definitions of commonly-required Sets of Attributes (currently 28 sets). Sets may include Attributes and other Sets in order to give human intelligibility to the complex collections of Attributes required to support business transactions; i.e. some Sets are logical groupings of information; others are structured to encapsulate business requirements. Examples of sets are ADDRESS, LOCATION, PERSON\_DETAILS, etc.

- Scenarios and Examples: By which the quality of support provided by the P/META Scheme for business transactions is being tested and thereby validated.

LSCOM

Recently, a new collaborative undertaking to develop and standardize the set of semantics for machine tagging of multimedia content, has emerged. The Large-Scale Concept Ontology for Multimedia (LSCOM) [179, 155], which is being led by IBM, Carnegie Mellon University, and Columbia University with participation from CyC corporation and various other research academic
and industrial groups, is sponsored by the Disruptive Technology Office (DTO) and consists out of a series of workshops that brought together experts from multiple communities. Their primary task was/is to create a taxonomy of 1,000 (or more) concepts for describing broadcast news video. The LSCOM taxonomy was designed to satisfy multiple criteria of utility, coverage, feasibility, and observability. Along with the taxonomy of 1,000 concepts, the LSCOM effort has produced a set of use cases and queries along with a large annotated data set of broadcast news video. The goal is to create a framework for ongoing research on semantic analysis of multimedia content, emphasizing on the automated extraction of describing terms. In comparison, for example, the MPEG-7 Genre Classification Scheme (urn:mpeg:mpeg7:cs:GenreCS:2001, which is a genre classification scheme defined by the MPEG-7 standard), which is used to classify programs based on their content or subject matter, defines terms such as “special events” and “remarkable people”. The terms might be useful for classifying multimedia content but do not lend themselves well to automated extraction. Such subjective concepts also make it difficult for two annotators to completely agree, which further complicates this issue. LSCOM was designed with the following requirements in mind:

- Utility: High practical relevance in supporting genuine use cases and queries.
- Coverage: High coverage of the overall semantic space of interest to end-users within the target domain.
- Feasibility: High likelihood of automated extraction giving a five-year technology horizon.
- Observability: High frequency of occurrence within video data sets from the target domain.

Furthermore, currently LSCOM is being mapped to the Cyc Knowledge Base, which is a repository of more than 300,000 concepts and 2 million assertions (rules and ground assertions). This mapping is intended to help ensure that the LSCOM ontology will be more than a taxonomy (simple hierarchy) of concepts. Cyc will provide a knowledge-rich representation, complete with rules and additional background knowledge which LSCOM needs to support a previously unattained level of semantic video and image annotation. Selected concepts were based on semi-automatic mapping of 26,377 noun search terms from BBC query logs in late 1998 to WordNet\textsuperscript{13} senses.

MOS

The Media Object Server Communications Protocol (MOS) is an evolving protocol for communications between Newsroom Computer Systems (NCS) and Media Object Servers (MOS) such as Video Servers, Audio Servers, Still Stores, and Character Generators \[163\]. This protocol is supported and developed through cooperative collaboration among equipment vendors, software vendors and end-users. While their main goal is to develop and maintain a common communications protocol which allows integration of diverse NCS and MOS equipment. It was developed by a group of software and hardware vendors in the Broadcast Industry late 1998 \[163\]. Some specific goals of the MOS group are (the whole list can be found in \[163\]):

- Develop and Implement a communication protocol for use between NCS and MOS which will allow these machines to communicate with each other independent of vendors.
- Common implementation of the protocol will be over a TCP/IP network via socket communication.
- Messages will be concise and optimized for speed of transmission and processing.
- Messages will make use of a tagged text unicode format.
- The protocol will be extensible to meet future industry requirements.

\textsuperscript{13}http://wordnet.princeton.edu/
With other words, the Media Object Server is responsible for the creation, modification, and deletion of media objects and their associated metadata. The Newsroom Computer System (NCS) is responsible for the creation, modification, and deletion of editorial information, including play lists. So in general, the NCS maintains the playlist defining the sequence of what will be put on broadcast, potentially containing a lot of assets like video, audio, pictures, etc. The MOS is the central serving system which contains all these assets and can make the NCS aware of its contents. During live production, the MOS protocol allows the playlist in the non-linear server to be changed in realtime by the newsroom computer system. A producer can resequence a rundown while the show is in production by simply moving stories at their workstation. Status information is passed in the background between the NCS and the MOS so the producer instantly knows what clips are ready and which are not.

The MOS Communications Protocol allows the NCS and MOS to exchange information using a standard protocol (language and vocabulary). This protocol enables the exchange of the following type of messages:

- Descriptive Data for Media Objects: The MOS pushes descriptive information and pointers to the NCS as objects are created, modified, or deleted in the MOS. This allows the NCS to be aware of the contents of the MOS and enables the NCS to perform searches on and manipulate the data the MOS has sent.

- Playlist Exchange: The NCS can build and transfer playlist information to the MOS. This allows the NCS to control the sequence that media objects are played or presented by the MOS.

- Status Exchange: The MOS can inform the NCS of the status of specific clips or the MOS system in general. The NCS can notify the MOS of the status of specific playlist items or running orders.

The MOS Protocol is fundamentally a tagged text data stream. In the version 2.x and later, data fields are character delimited using XML tags defined in the MOS Data Type Definition (DTD). In MOS v1.x data fields were delimited using a proprietary format.

In MOS there is a large number of messages defined to control the flow of data between the systems. While the complete list of messages can be found in [164], we give the example of a roItemStat message in Figure 2.10 which is responsible for updating the NCS on the status of an individual Item in a Running Order. This allows the NCS to reflect the status of individual Items in the MOS Running Order in the NCS Running Order display.

```xml
<mos>
  <mosID>aircache.newscenter.com</mosID>
  <ncsID>ncs.newscenter.com</ncsID>
  <messageID>506702</messageID>
  <roItemStat>
    <roID>5PM</roID>
    <storyID>HÔTEL FIRE</storyID>
    <itemID>0</itemID>
    <objID>A0295</objID>
    <itemChannel>B</itemChannel>
    <status>PLAY</status>
    <time>1999-04-11T14:13:53</time>
  </roItemStat>
</mos>
```

Figure 2.10: Example of a MOS message
2.5. MODELING THE DOMAIN

MXF

MXF is a file format for the exchange of program material between servers, tape streamers and to digital archives [192]. Its contents may be a complete program as well as complete packages or sequences. There are basic facilities available for cuts between sequences and audio cross-fades. This way the sequences can be assembled into programs. MXF is self-contained, holding complete content without need of external material.

MXF bundles together video, audio, and program data, such as text (together termed essence) along with metadata and places them into a wrapper. Its body is stream based and carries the essence and some of the metadata. It holds a sequence of video frames, each complete with associated audio, and data essence, plus frame-based metadata. The latter typically comprises timecode and file format information for each of the video frames. This arrangement is also known as an interleaved media file.

The body can be based on several different types of material (essence) including MPEG, DV and uncompressed video and audio, it also uses the SMPTE KLV data coding system, which has the advantage of being a recognized standard. The development of the Material Exchange Format (MXF) is a remarkable achievement of collaboration between manufacturers and between major organizations such as Pro-MPEG [192], the EBU [87] and the AAF Association [1]. It establishes interoperability of content between various applications used in the television production chain. This leads to operational efficiency and creative freedom through a unified networked environment.

MXF is not compression-scheme specific and simplifies the integration of systems using MPEG and DV as well as future, as yet unspecified, compression strategies. This means the transportation of these different files will be independent of content, not dictating the use of specific manufacturers’ equipment. A major aim of MXF is the seamless passage of program content and its associated metadata. The problem is that, due to incompatibilities, this information is currently lost as the content moves between systems. MXF-enabled systems will communicate using metadata, video and audio. MXF metadata may carry information about e.g. the file structure, the body contents e.g. MPEG or DV, keywords or titles, subtitles, editing notes, location, time, date and etc. MXF wraps all this various kinds of metadata (it does not restrict which metadata) to be able to send it along with the multimedia stream in a structured way.

MPEG-21

MPEG-21 aims at defining a normative open framework for multimedia delivery and consumption for use by all the players in the delivery and consumption chain [175]. MPEG-21 is based on two essential concepts: the definition of a fundamental unit of distribution and transaction (the Digital Item) and the concept of Users interacting with Digital Items. The Digital Items can be considered the ‘what’ of the Multimedia Framework (e.g. a video collection, a music album, etc.) and the Users can be considered the ‘who’ of the Multimedia Framework. The goal of MPEG-21 can thus be rephrased to: defining the technology needed to support Users to exchange, access, consume, trade and otherwise manipulate Digital Items in an efficient, transparent and interoperable way.

The MPEG-21 specification (also made by the Moving Picture Experts Group [177]), basically consists out of nine main parts. Besides the more general parts like: vision and strategy, Digital Item Declaration (DID), digital item identification, software and the file formats, the main parts primarily focus on the intellectual property and rights management. MPEG-21 defines an interoperable framework for Intellectual Property Management and Protection (IPMP). However, fairly soon after MPEG-4, with its IPMP hooks, became an International Standard, concerns were voiced within MPEG that many similar devices and players might be built by different manufacturers, all MPEG-4, but many of them not interworking. This is why MPEG decided to start a new project on more interoperable IPMP systems and tools. Here it addresses authentication of IPMP tools, and has provisions for integrating Rights Expressions according to the Rights Data Dictionary and the Rights Expression Language (REL). This REL standard is used as the means of sharing digital rights/permissionsrestrictions for digital content from content creator to content consumer.
Dublin Core

The Dublin Core Metadata Initiative (DCMI) is an organization engaged in the development of interoperable online metadata standards and vocabularies that support a broad range of purposes and business models [82]. DCMI currently focuses on architecture and modeling, discussions and collaborative work in DCMI Communities and DCMI Task Groups to promote widespread acceptance of metadata standards and practices. It is their mission to provide simple standards to facilitate the finding, sharing and management of information. The ‘Dublin’ in Dublin Core refers to the first workshop, which was held in Dublin, Ohio.

One of the main activities of DCMI is the development and maintenance of a core set of metadata terms. This Dublin Core metadata standard is an ANSI-approved, internationally accepted specification for describing electronic information. It includes fifteen data elements for describing and organizing electronic resources to make them easier to find and access over the Web. Dublin Core elements can be extended to meet the special needs of individual disciplines while promoting interoperability among disciplines. A significant and growing number of people, organizations and governments use the Dublin Core standard and participate in DCMI activities. Initially, Dublin Core was designed specifically for generating metadata for textual documents [130]. After a number of workshops held to discuss the applicability of Dublin Core to non-textual documents such as images, sound and moving images, they extended the 15 core elements through the use of sub-elements and schemes specific to audiovisual data.

While being a simple yet effective element set for describing a wide range of networked resources. The Dublin Core standard includes two levels: Simple and Qualified. Simple Dublin Core comprises fifteen elements (among which there are: Title, Creator, Subject, Description, Publisher, Date, Type, etc.); Qualified Dublin Core includes three additional elements (Audience, Provenance and RightsHolder), as well as a group of element refinements (also called qualifiers) that refine the semantics of the elements in ways that may be useful in resource discovery. The semantics of Dublin Core have been established by an international, cross-disciplinary group of professionals from librarianship, computer science, text encoding, the museum community, and other related fields of scholarship and practice. In this language, there are two classes of terms, elements (nouns) and qualifiers (adjectives), which can be arranged into a simple pattern of statements. The resources themselves are the implied subjects in this language. In the diverse world of the Internet, Dublin Core can be seen as a “metadata pidgin for digital tourists”, easily grasped, but not necessarily up to the task of expressing complex relationships or concepts.

Three Dublin Core principles bear mentioning here, as they are critical to understanding how to think about the relationship of metadata to the underlying resources they describe:

- The One-to-One Principle: Dublin Core metadata specifies one description or version for every resource, rather than assuming that descriptions stand in for one another.
- The Dumb-down Principle: The qualification of Dublin Core properties is guided by a rule known colloquially as the Dumb-Down Principle. According to this rule, a client should be able to ignore any qualifier and use the value as if it were unqualified. While this may result in some loss of specificity, the remaining element value must continue to be generally correct and useful for discovery.
- Appropriate values: Best practice for a particular element or qualifier may vary by context, but in general an implementor cannot predict that the interpreter of the metadata will always be a machine.

Limited by the fifteen concepts in the Simple Dublin Core, qualifiers (extensions) can be added to have a more specific and fine-grained set of properties to describe resources. In [130], Hunter proposes an extension to Dublin Core for Moving Images, including among others properties like Format.video.codec, Format.video.framerate, etc. The downside of the ability to add your own set of semantic refinements, is the potential loss of consistency and interoperability. In Figure 2.11 we see an example (extracted from [130]) of some Dublin Core (DC) properties in a RDF description of a movie resource.
2.5. MODELING THE DOMAIN

AAF

The Advanced Authoring Format (AAF) is being developed and promoted by the AAF association to enable content creators to easily exchange digital media and metadata across platforms, and between applications [1]. AAF is an industry-driven, cross-platform file format that allows the interchange of data between multimedia authoring tools. AAF can be used to interchange data (like picture, sound and other forms of data that can be directly perceived) and metadata.

To allow for more complex functionality, AAF provides an object model to describe the data and metadata. The structured approach of the object model makes it easier to describe complex data and allows for simple interchange between parties. The AAF object model basically provides the following capabilities:

- A mechanism to encapsulate data and metadata, which allows an application to determine the format and, if any, the conversions it needs to apply to process it.
- A mechanism to synchronize data and to describe interleaved streams, allowing an application to integrate separate streams of data that were created in synchronization.
- A mechanism to describe derivations from the original media sources.
- A mechanism to describe compositions. Compositions contain information about how sections of data should be combined in sequence, how to synchronize parallel tracks of sequences, and how to alter or combine sections by performing effects.
- A mechanism to define new classes or to add optional information to existing classes, allowing applications to store additional information in an interchangeable way.

The central AAF object is a package. A package is an object that has a universal (globally unique) identifier and consists of metadata. Packages have names and descriptions, but are primar-
ily identified by a unique identifier, which is called a PackageID. There are four kinds of packages that are commonly used in the AAF object model, namely composition packages (combination of data with specific order, placements and effects), material packages (access and synchronization of the data independent of the storage details), file source packages (media stored in a file) and physical source packages (describes media on e.g. video, CD, etc.).

A package can describe more than one data element. For example, a package can have audio, video, still image and timecode data. Each one of those elements in a package is called a slot. Each slot can describe only one type of data, and it can be referenced from outside of the package. Each slot in a package has a SlotID that is unique within the package. To reference the data in a slot, the PackageID and the SlotID are used. Three kinds of slots are commonly used in the AAF object model: a static slot (no specific relationship with time), a timeline slot (a continuous relationship with time) and an event slot (a irregular relationship with respect to time). Furthermore, a slot contains a segment to further describe its data element [2].

2.5.3 Comparison

Looking at this list of standards and specifications, mainly created over the last decade, it is clear that many organizations, associations, groups and fora see the need for a well-structured metadata standard, enabling the description, sharing and integration of various audiovisual content elements. In the field of audiovisual content, this urge for standardization is mainly driven by the need for interoperability which is indispensable considering the large number of different parties working in various niches of the content production/broadcasting chain.

In the effort to find an answer to our research question, we need to investigate current data specifications in the TV domain suitable to support interactive and personalized television applications. Obviously, our eventual choice of a data specification to model television content, will have a profound impact on every subsequent step later in the data manipulation process (including for example integration and personalization), indicating that this choice should be taken with care. However, more options does not make the choice any easier. In the extreme case to fulfill our requirements, creating yet another specification might prove necessary.

While most of the presented specifications facilitate straightforward multimedia features like title, author/creator, date of creation, etc. we are more interested in the underlying reflections and considerations made when they were modeled. Each of these specifications was modeled with a specific problem in mind that needed to be solved. TV-Anytime for example, was specifically designed to enhance navigation and search in consumer television applications like program guides, while MXF was optimized for the interchange of material between the content creation industries.

To be able to compare the specifications previously discussed, we rely on the categorization proposed in [218], which was also used in other multimedia specification overviews like [167]. In [218], Smith et al. argues that despite various metadata standardization efforts, several reasons exist why creating order and overview is still elusive. However, these reasons have little to do with metadata’s value. Mostly, in the situations they were built for, existing metadata practices or standards provide enormous value. Further, Smith et al. retrace that the problem and complexity of metadata standardization efforts results from three primary difficulties. Namely, the what, how and why questions. Every standard is orientated towards a specific goal which might have different answers to these questions. Therefore, afterwards, it can prove very difficult to reconcile different standards. However, investigating these different levels of design more closely, can help us in choosing and estimating which standard can prove most useful to satisfy our problem description.

The different facets in the production/broadcasting chains are so divers that it becomes impossible to model one general standard that solves it all. Therefore, in the ‘what’ question, we take a closer look at the goal and what the specification is supposed to describe. Smith proposes four distinct dimensions to describe a standard’s domain and goal:
2.5. MODELING THE DOMAIN

• Industries: E.g. music, film, TV, enterprise, etc., the branch of productive (commercial) usage.

• Processes and workflows: E.g. creation, production, management, distribution, etc., understood in terms of the Canonical Processes of Media Production [113].

• Content types: E.g. images, video, audio, etc., or types of media the metadata is capable to describe.

• Content domains: E.g. news, sports, entertainment, etc., the realm in which the standard is intended to be used.

In Table 2.1, we see the previously discussed standards classified based on these dimensions. As shown, some standards (e.g. AAF, SMPTE MD, MXF, etc.) were developed to support content producers and provide them with means to describe data usually in a more technical manner. Others like MPEG-7 and Dublin Core, are more general purpose descriptions (Dublin Core even more than MPEG-7) which are able to describe any kind of multimedia and can therefore be utilized in various parts of the production/broadcast chains. Further, specifications like TV-Anytime, XML-TV, Escort and ProgramGuideML particularly focus on TV programs, making themselves very suitable for the publishing of program metadata. XML-TV is a special case, since it grew out of a community of people who privately agreed to model a specification serving their program description needs in feeding various EPG systems.

The ‘how’ question tries to clarify the way in which the standard is modeled. Through the years, many groups started to define metadata specifications. However, this resulted in a rather unstructured approach to the problem. As described by Smith, nowadays there are multiple metadata representations, encodings and levels:

• Domain modeling: Defines concepts and meanings for the target industries, workflows and processes, content types, and content domains.

• Metadata scheme representation: Gives the domain concepts generally in the form of entities, attributes, and relationships.

• Schema definition: Represents the schema in a particular syntax, such as XML Schema

• Semantics representation: Explains the content’s representation style for example, textual description

• Metadata encoding: Encodes the metadata as bitstream representation for example, as XML, binary XML, or SMPTE key-length-value (KLV) encoding.

While any of these ways of representation serve a specific goal, it is less of an issue for us personally. A good representation is important but it does not directly influence the functionality or features of the system. At most it will have an influence on the performance or processing overhead. Furthermore, most of the discussed standards use a XML Schema or DTD definition.

Lastly, the ‘why’ question, tries to find an answer to: “for which purpose was this specification built?”. There are several possibilities and moreover, one single specification can serve multiple purposes. Smith chose six central purposes: search, summarization, repurposing, personalization, adaptation and exchange/transport. While a more generic standard like MPEG-7 or Dublin Core, can be used to serve several, others like e.g. MXF are built for one specific purpose (in this case exchange/transport). However, besides the six categories defined by Smith, another important perpendicular aspect of the specification’s ‘purpose’ involves its business type. Some standards are specifically intended to be used between businesses (B2B) while others focus on customer services (B2C). This aspect is also important with regard to our research question, since we are interested in specifications to support interactive and personalized television applications to the user. In Table 2.2, we see an overview of the previously discussed specifications in terms of their purposes.
<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Industry</th>
<th>Workflow</th>
<th>Content type</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-7</td>
<td>non-XML, XML</td>
<td>generic</td>
<td>publishing</td>
<td>audio, video, images</td>
<td>generic</td>
</tr>
<tr>
<td>Escort</td>
<td>non-XML</td>
<td>broadcast</td>
<td>publishing, administration</td>
<td>audio, video</td>
<td>radio- and TV programs</td>
</tr>
<tr>
<td>TV-Anytime</td>
<td>XML</td>
<td>broadcast</td>
<td>distribute, publishing</td>
<td>general purpose</td>
<td>TV programs</td>
</tr>
<tr>
<td>XML-TV</td>
<td>XML</td>
<td>private</td>
<td>publishing</td>
<td>audio, video</td>
<td>TV programs</td>
</tr>
<tr>
<td>Program GuideML</td>
<td>XML</td>
<td>program guides</td>
<td>publishing</td>
<td>general purpose</td>
<td>TV programs</td>
</tr>
<tr>
<td>SMPTE MD</td>
<td>non-XML, XML</td>
<td>broadcast</td>
<td>production</td>
<td>audio, video</td>
<td>content creation</td>
</tr>
<tr>
<td>EBU P/Meta</td>
<td>XML</td>
<td>broadcast</td>
<td>distribution, publishing</td>
<td>audio, video</td>
<td>generic</td>
</tr>
<tr>
<td>LSCOM</td>
<td>non-XML, OWL</td>
<td>broadcast</td>
<td>publishing</td>
<td>audio, video</td>
<td>news video</td>
</tr>
<tr>
<td>MOS</td>
<td>non-XML, XML</td>
<td>broadcast</td>
<td>production, distribution, publishing</td>
<td>audio, video, images</td>
<td>newsroom</td>
</tr>
<tr>
<td>MXF</td>
<td>non-XML</td>
<td>broadcast</td>
<td>production</td>
<td>audio, video, images</td>
<td>content creation</td>
</tr>
<tr>
<td>Dublin Core</td>
<td>XML, RDF</td>
<td>generic</td>
<td>publishing</td>
<td>general purpose</td>
<td>generic</td>
</tr>
<tr>
<td>MPEG-21</td>
<td>non-XML, XML</td>
<td>generic</td>
<td>annotate, publishing, distribute</td>
<td>general purpose</td>
<td>generic</td>
</tr>
<tr>
<td>AAF</td>
<td>non-XML</td>
<td>broadcast</td>
<td>production</td>
<td>audio, video, images</td>
<td>content creation</td>
</tr>
</tbody>
</table>

Table 2.1: Audiovisual metadata specification overview
2.5. MODELING THE DOMAIN

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
<th>Business Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-7</td>
<td>search, personalization, adaptation</td>
<td>B2C</td>
</tr>
<tr>
<td>Escort</td>
<td>search, exchange/transport</td>
<td>B2C</td>
</tr>
<tr>
<td>TV-Anytime</td>
<td>search, personalization, adaptation</td>
<td>B2C</td>
</tr>
<tr>
<td>XML-TV</td>
<td>search</td>
<td>B2C</td>
</tr>
<tr>
<td>ProgramGuideML</td>
<td>repurposing</td>
<td>B2C</td>
</tr>
<tr>
<td>SMPTE MD</td>
<td>summarization, exchange/transport</td>
<td>B2B</td>
</tr>
<tr>
<td>EBU P/Meta</td>
<td>exchange/transport</td>
<td>B2B</td>
</tr>
<tr>
<td>LSCOM</td>
<td>search, summarization</td>
<td>B2C</td>
</tr>
<tr>
<td>MOS</td>
<td>repurposing, exchange/transport</td>
<td>B2B</td>
</tr>
<tr>
<td>MXF</td>
<td>exchange/transport</td>
<td>B2B</td>
</tr>
<tr>
<td>Dublin Core</td>
<td>search, summarization</td>
<td>B2C</td>
</tr>
<tr>
<td>MPEG-21</td>
<td>search, exchange/transport</td>
<td>B2C</td>
</tr>
<tr>
<td>AAF</td>
<td>repurposing, exchange/transport</td>
<td>B2B</td>
</tr>
</tbody>
</table>

Table 2.2: Audiovisual metadata specification purposes

As expected, from Table 2.2 we can deduce that most B2C specifications focus on customer-related activities like providing ‘search’ while most B2B standards concentrate on the exchange and transport of data between companies and/or different formats. In Figure 2.12, we see on the left hand side all metadata specifications which are primarily built for B2B (Business-2-Business) purposes. For example, the MOS (Media Object Server communication protocol) is built for communication between a Newsroom Computer System (NCS) and a Media Object Server e.g. serving video and audio streams. MXF for example focusses on the exchange between servers, tape streamers and digital archives [192]. In the specifications of these standards we thus mainly find properties describing the type of production (e.g. audio mode), status, administrational data (e.g. copyright owners), organizational data, etc. With respect to our research question, none of them really fit to support interactive and personalized television applications which could require properties like e.g. titles, synopses, genres, credits, etc.

On the right hand side of Figure 2.12 we see the metadata specifications primarily built for B2C (Business-2-Customer) purposes. In contrast to the B2B solutions, these are specifically built for the distribution and exploitation of audiovisual content to the end-user. As seen in Figure 2.12, we have split the group of B2B standards in one group specifically made to model radio and television programs and a second group containing specifications to model audiovisual data in general. Do note that Dublin Core is even more generic since it was created to support a broad range of purposes and therefore able to also model other kinds of objects like e.g. books, publications, etc. Since these more general specifications are not specifically tailored for the television domain, they therefore have difficulties to describe concepts like e.g. a broadcast channel, program participators like an actor, director, presenter, etc. The specific television program metadata specifications, including Escort 2007, XML-TV, LSCOM, ProgramGuideML and TV-Anytime, are superior in this sense and therefore good candidates to describe television program metadata.

2.5.4 Conclusion

Many different metadata specifications and standards for audiovisual content are described over the years. For some reason, developing a custom description usually gets more support than reusing an existing one. As Murtha Baca from the Getty Vocabulary Program once putted it: “Standards are like toothbrushes; everyone agrees they are a good idea, but nobody wants to use anyone else’s”. However, all of these specifications were built for specific goals, in various formats and with different semantics. Therefore, not every specification is the optimal choice for a particular situation. On
the other hand, the best choice technically, is not always the most practical one. Imagine a system where for example lots of data are retrieved from external sources and enriched before being sent to a consumer device. There, it can be more beneficial to choose a metadata specification which is the same or similar to the format used by the purveyor, rather than to take the “on paper” best specification for the task.

Choosing the right specification for the task depends on many factors. The most obvious comes from the fact that specifications usually target either B2B (Business-2-Business) or B2C (Business-2-Consumer) applications. Further, studying the ‘what’, ‘how’ and ‘why’ of the problem also results in considerable down-scaling of the potential candidates. Furthermore, the choice can be heavily influenced by some more pragmatic considerations. Deciding to use one of the more well-known and respected Internet standards, can turn out very beneficial in terms of experiences and tool support. In case of extensive computer processing of the metadata, choosing for a well-known language like XML can save time and effort as many people know it, and problems can be solved easily.

Which specification of this selection of five will prove best for our particular situation, will be shown in the next chapter where we take a closer look at the requirements of the system envisioned.

2.6 Conclusion

In this chapter we gave an introduction to digital television with an emphasis on the evolution throughout the last decade. As shown, interactive television is currently still very limited, yet having a potentially bright future in prospect. Considering the remarkable rise of the Internet over the years, it is not unthinkable a similar path lies ahead for the television platform as well. The growth of the Internet was mainly driven by the subconscious human need to communicate with one another and find kindred spirits around. Similarly, the adoption of the Internet connection on set-top boxes, thereby enabling a necessary return channel, will facilitate the same expansion of interactivity on the TV platform.
2.6. CONCLUSION

By looking at the state-of-the-art in the set-top box world, we have elucidated current possibilities and shortcomings presently suffered from. On the positive side, we saw that most set-top boxes are equipped with universally known middleware solutions like OpenTV or MHP so that different boxes can run all software installments made for their type of middleware. However, the main restrictions like e.g. rarely having a return channel, the lack of local storage and the limited hardware specifications, severely limit the capabilities and potential functionalities. However, we are inclined to say that with recent market developments, also the set-top box hardware specifications will be extended quickly as the requirements of the market increase.

When potentially thousands of Web channels are unlocked and people can start sharing their homemade content, the information overload will make general zapping inadequate for navigation and searching of content. Therefore we are striving for an approach which will enable the user to find exactly that content he or she is looking for at any particular moment. However, this functionality depends heavily on the quality of the metadata descriptions of content elements. Therefore, in this chapter, we gave an extensive overview of existing audiovisual metadata specifications together with their specific goals and structure.

In this chapter we have answered research question 1 (Which technologies and standards, prevalent in the television domain, exist and can be suitable to support interactive and personalized television applications?). We have introduced the television domain, and gave an extensive overview of technologies and standards which can be utilized to support our goal. However, before we can make a clear choice concerning which technologies and metadata specifications we can use, we need to take a deeper look at the requirements of interactive and personalized television applications.
Chapter 3

Requirements

Finding the right information on the Web is still a tedious task. Partially this is due to the large amount of data around, but mainly because this information is usually not very well-structured nor annotated. Search engines like Google find relatively good query answers via the brute force method: going through all available Web sites to see whether they contain the words of the query, and return the results ordered by means of a number of parameters (see the page rank algorithm introduced in [50]). While this approach works perfectly given the circumstantial difficulties, the size and numbers of the Google server parks across the globe tell us a different story. Still, despite this immense calculation monster, advanced queries like e.g. “What do the Niagara falls look like at twilight in January from the Canadian side?” remain unanswered, not counting the millions of partially related results returned.

In an attempt not to make the same mistakes twice, a strong tendency is rising where people start thinking about how to model various kinds of objects like e.g. trees, cars, movies, people, etc. appropriately. After all, it is the model which contains the relations, semantics and structure which will later be the backbone of any functionality or feature working on top of these objects. Good modeling now, will definitely avoid annoyances later. However, if you let several people each individually model a set of classes in a specific domain, every one of them will probably come up with a different and original approach. It has occurred more than once that two different groups model the same domain, each starting from their own viewpoint and requirements, and end up with two completely different approaches. However after some time, those groups learn about each other and decide that it would be nice to make their individual models compatible to increase interoperability. Usually, this then proves to be a big challenge as tiny little differences in semantics can cause insurmountable problems in defining the optimal alignments. Therefore, also here the cliché “Stand on the shoulders of giants” is very relevant. If there already exists some generally community-accepted model which suits the larger part of your requirements, it could be better to adapt or extend this model, than starting completely from scratch by creating yet another new one.

In this chapter, we define various terms which are used throughout this dissertation. Further, we discuss the requirements of our goal with respect to the state-of-the-art defined in Chapter 2. This chapter addresses research question 2 (What are the requirements for an approach providing user-adapted data retrieval?).

3.1 Introduction

Choosing or developing an appropriate model for any specific domain is not an easy task. In the previous chapter for example we already saw an overview of the multitude of different audiovisual description schemes which currently exist. However, it was clear that each of those schemas was tailored with a specific goal or set of requirements in mind. Moreover, we could even notice particularly foci on for example different parts in the content production/broadcasting chain or on
the interoperability between different parts in this chain. In this chapter we look at our requirements and investigate which television content description schemes might meet these requirements. While it is our preference to use, and if necessary extend, an existing description standard, if none satisfies our requirements it might be unavoidable to create a new one ourselves. However, before we can define our requirements, we first introduce some general as well as television specific modeling terms used throughout this dissertation.

### 3.1.1 Some definitions

There exist various terms to talk about digital bits and bytes. Data, information, model, resource, knowledge, etc. are all terms which are often used in the domain of computer science. While being very abstract concepts usually without some straightforward definitions available, we will try to define here how these terms are meant to be interpreted throughout the text.

According to Wikipedia, ‘data’ is often viewed as the lowest level of abstraction. Everything we have, know or obtain describing any other thing can be considered as data. As an analogy, we can compare data with all the matter in the universe. It resides all around us, behaves chaotically and incoherent, but the size and dimensions are overwhelming. ‘Information’ on the other hand, can be considered as a subset of the data which is somewhat interesting to filter out as it differentiates itself from the rest by sharing some specific attributes. E.g. all the data which is related to object A can be seen as the information we have about A. In the analogy, it would exemplify a part of the universe which together identifies some specific part, like for example a galaxy. ‘Knowledge’ can be interpreted as obtained insights and expertise which resulted from thorough investigations of that information. Analogously, after studying and observing matter in the universe, the insight is formed that gravity is responsible for the formation of clusters and galaxies. ‘Knowledge’ is then the conception of how and when this phenomenon can occur.

Besides data in general, there are also terms to model a specific entity in a larger set of things. On the very general level, we have a ‘resource’ which represents any physical or virtual entity. While the term ‘object’ is very domain dependent, in computer science an object is very comparable to a resource. According to Wikipedia an object is a compilation of a set of attributes with a determined behavior. A ‘class’ on the other hand, is the general name of a group of objects with common attributes, characteristics and behavior. In our analogy, a class can denote everything from stars, planets, nebulae, asteroids, comets, etc., while resources and objects also include the specific members of those classes like e.g. planet earth, the sun, Halley’s Comet, the Horsehead Nebula, etc. Similarly, the term ‘instance’ is also often used to denote a real existing occurrence pertaining to a class. However, an instance is only meaningful in relation to its class. While everything can be called a resource or object, an instance must belong to a well-defined class. E.g. every existing planet in the universe is an instance of the class ‘planet’. Without the definition of that class, it remains just an object or resource.

Another often used term in computer science, is the term ‘concept’. The difference between terms like ‘resource’, ‘object’, ‘class’, etc. on the one hand and a ‘concept’ on the other hand however, is less clear. It has been described by the philosopher Gottlob Frege at the end of the 19th century [96], however, in this dissertation we will use the term ‘concept’ as a synonym for a ‘class’. ‘Content’ is a term which is very domain specific as it denotes the central subject of the matter. In this dissertation, when talking about ‘content’ we actually refer to the audiovisual data like for example a program, movie or music itself which was described in the metadata. A “content-element” therefore denotes one specific element of content.

When all objects and relations are defined within a specific domain, they together form a domain model. Such a domain model thus contains all the classes to model the entities appearing in that domain together with all relationships between those entities. Besides the term “domain model”, we can also use “conceptual model” or “data model”. Strictly speaking these terms are broader as they are not limited by any specific domain. We can therefore see a domain model as a conceptual model or data model valid within a specific domain. In this thesis, considering our illustrative domain, we interpret these three terms as synonyms. Similarly to the model grasping the semantics of the central domain, a user model or user profile models the user or person
interacting with the system. It models for example all the user’s characteristics, demographics, interests, etc. with respect to the central domain. However, considering the slight conceptual difference between a user model and a user profile, which we explain later, for now we just use the former.

In the audiovisual domain there are a number of concepts specific to this domain which are used throughout this dissertation. Every individual entity which can be broadcast or displayed through any form of distribution channel, independent of its type (movie, episode, etc.) or source (broadcaster, Web, VOD, etc.), we call a program. To classify programs in terms of their contents, we use two orthogonal dimensions: format and genre. The format deals with the structure of the program regardless of the subject, while the subject itself is captured by the genre. A program can possibly be a well-defined numbered issue in a series, which makes it an episode of that series. Similarly, a program can also be an unnumbered part in an endless succession of programs in a series, which makes it a series’ member. Sometimes a program can be released in slightly different incarnations of the same program (e.g. the 8 o’clock news and the 10 o’clock news both present “the news”), which make it different editions.

### 3.2 Requirements

The main research question of this dissertation deals with how we can provide context-sensitive and user-adapted access to large heterogeneous data sources. In other words, if we have a user, from whom we have some kind of description showing what kind of a person he or she is, and we have a large pile of data accompanied by some form of metadata describing it, we want to figure out how we can help this user to find exactly that subset of information he or she is most interested in at this particular point in time, regardless of the domain.

Towards this goal, this dissertation describes an approach which we illustrate within the television domain. More broadly, we investigate the design of a ubiquitous home media architecture of connected devices that can provide the user with access to a wide range of media sources, yet at the same time avoid an overflow of information. Our implementation framework consists of the central iFanzy server framework and three iFanzy applications (running on top of the server and further explained in Chapter 9), aiming to connect different devices, such as shared (large) screens with set-top boxes, personal (small) hand-held devices and biosensor-based interfaces, and different media sources like IP, broadcast and local storage. This intentionally goes beyond the traditional limited solution of a single TV screen and simple remote control and therefore creates the foundation for an ambient home environment to collect various data about the users and to subsequently use this data for the personalization of his/her interaction with the TV content. From this description we can distill three major areas of requirements necessary to reach this goal: requirements with respect to the domain model, the user model and the adaptation process. The main requirements for each of these are explained in further detail in the next subsections.

#### 3.2.1 Domain model

A first interesting set of requirements deals with the domain model and how to enable intelligent access to the data it describes. In Chapter 2, we already saw that rich metadata plays a huge role in this process, as it allows us to assess and maximally utilize the data. The more and richer metadata we have the better we can compare key properties, filter out specific items and interpret the user’s feedback. Rich metadata however, does not come for free. Firstly, a metadata specification must allow richness by having a wide diversity of properties, a well-defined semantic structure and allow for possible later extensions. Secondly, a rich specification can only yield results if for every content-element meaningful and rich metadata is gathered, either automatically or manually. Besides richness, identifiability can be considered equally important. In our data model every object should be identifiable by means of a unique identifier to allow for unambiguous referencing.
Since the data model defines how we can describe relevant domain items, its properties define what we will be able to express. Therefore, it is important to accept and understand the world around and allow for connections to other already existing resources. There already exist for example well-defined term hierarchies which can be used to classify objects (e.g. as described in Chapter 2). These hierarchies make a good and universally accepted addition to general resource descriptions. Further, the ability to refer to related content like e.g. pictures, other programs, books, etc. improves the expressiveness of the model, as well as navigation across the domain. Moreover, the specification should support the reuse of existing semantics as much as possible.

Further, the last few years on the Internet we all witnessed the rising importance of the integration of large data sources. With the release of various APIs we see for example that external sources like Wikipedia, Flickr, Facebook, etc. are commonly referred to. This logical evolution entails relative simple access to large amounts of invaluable information relevant in various domains. Hence, it makes more sense to utilize these specialized sources than to try and create something similar yourselves. With an eye on the future, one of the most important requirements of our domain model should therefore be that integration with other sources should be an easy thing to do. Moreover, when bringing together data from different sources, we must consider that these statements can together lead to the discovery of new facts through logical reasoning. Therefore, our data model must allow reasoning engines to run and discover such facts.

A further requirement involves that the model should, to some extent, be future-proof. In the previous chapter we saw that we are standing at the verge of a television revolution, which might introduce new concepts like Web channels, user generated content, etc. Therefore it is essential that our domain model is flexible enough to be extended also along these lines. People will at one point for example be able to annotate their own content themselves which will unavoidably introduce new requirements to the domain model. The domain in any way should be suitable for various media coming from the Web, portable media (CD, DVD, Blu-Ray, etc.), local storage, etc. Another future functionality relates for example to the ability to describe not just programs, but also segments within programs. This would enable to specifically describe the segment of for example every goal in a soccer match.

A last requirement for the domain model concerns the performance of the system. Large data collections can potentially put a constraint on the performance of the system. Therefore, the model should avoid too cumbersome or exotic class constructions as it can lead to complex querying afterwards. The domain should be modeled in a straightforward way such that the most frequently asked data sets can be return responsively.

### 3.2.2 User model

One of the most essential requirements to be able to automatically adapt any kind of functionality to a person is the availability of an extensive and computer-understandable model describing this person. Such a model would hold potentially all the facts, feedback and behavior that the user sends out to the world, summarized and stored in one consistent container. Such a model, describing the user digitally to the largest possible extent, we call the user model. Moreover, to gather user data as unobtrusively as possible, the model must consider, besides explicit feedback, also implicit feedback reflecting the user behavior at any time.

With respect to the structure of the user model, our requirements are very similar to those of the domain model. The user model should be flexible such that it can easily be extended, it should be generic enough to serve various purposes or even be used in different domains or applications, it should allow for the integration of data from other external data sources and the structure should support reasoning over its data to be able to deduce new statements.

Opposed to the domain model, where the instances are gathered by means of automatic processes, the largest part of user model data is usually coming from the user himself. However, avoiding the situation where just anything is added to the user model, possibly leading to an over-complicated and oversized repository, we need to control the flow of new information. Therefore, extra requirements apply when new user feedback is obtained. We require that user data is parsed in a controlled fashion, such that only relevant information is effectually being added.
Modeling a user is far more complex than modeling any type of static object, like for example a TV program, in terms of subjectiveness, validity and context of facts. This means that at any time it is important to know in which specific circumstances a particular user statement is valid. E.g. it is not because a user likes action movies a lot, that he would like to watch The Terminator at 8 o’clock in the morning. Hence, it is important to know that this user only likes action movies on for example a Saturday evening. Moreover, our illustrative domain is a very context-sensitive one. The user’s mood, companions, current time, devices, etc. all influence his or her current television preferences considerably. Therefore, it is important for the user model to cater for a structure which can model such information.

Further, almost every user-adapted architecture runs into problems when new users subscribe to the system while still having an empty user model. Hence, it is hard to adapt any kind of data processing when nothing is known of the user in question. This problem is known as the cold start problem. In an effort to alleviate this problem as much as possible we require from the user model to have some built-in measures to minimize consequences for the user.

Besides the fact that we require an extensive, context-sensitive user model, it is also necessary to have a user model which is structured ‘universally’. With this we mean, that it should follow best practices such that it can easily be reused, increase interoperability with other approaches and utilize proven technologies and concepts, independent of the domain.

Lastly, we require a close correspondence between the user model on one side and the data model on the other. Through the previously stated requirement to have the ability to uniquely reference every resource in the data model, every user model entry should convey the user’s reflections on those domain items solely by using these unique references.

### 3.2.3 Adaptation

The term ‘adaptation’ refers to a process, active in an interactive system, adapting its behavior to individual users based on information acquired about its user(s) and its environment [252]. Therefore, in theory adaptation can be applied in any part of the data processing pipeline. Practically we already witnessed many different manifestations of adaption through various existing applications and Web sites. Adaptation can among others occur in technologies like:

- **Navigation**: Helping the user in finding the optimal path through the maze of information. E.g. hiding or recommending links in a Web page, ordering results based on previous experiences, etc.

- **Presentation**: Adaptation of the presentation of content to match the way the user likes it most. E.g. the theme of the interface can differ per user.

- **Data retrieval**: Adapting the data retrieval process such that the user retrieves the items he appreciates most. E.g. inappropriate results can be filtered out.

- **Consumption**: Similarly to adapting the content’s presentation, we can help the user further while consuming content. E.g. automatically adapting the program’s subtitle language to his or her mother tongue, while watching a program some people prefer three short advertisement breaks while others might favor one long break, etc.

- **Advertisement**: Adaptation of advertisements such that different users get different products advertised matching their personal interests.

Firstly, we require that adaptations such as shown in this list can also be applied to a group of people. After all, this is very relevant in the television domain as people often watch programs together. In such a case where people share the experience as a group, the interests and preferences of all need to be taken into account. E.g. subtitles shown in a program are adapted to the language understood by the largest subgroup. Secondly, as previously introduced adaptation can also depend on the current situation, commonly known as context. Therefore, we require that
CHAPTER 3. REQUIREMENTS

the adaptation processes take such information into account. E.g. the interface’s theme can differ with the time of the day, advertisements change based on the current device, etc.

Naturally, in order to provide an adapted service to the user in a particular domain, the adaptation process is very much dependent on both the richness of the relevant item descriptions in the domain, as well as on the quality of the user model. Therefore, the two main requirements for the adaptation process are on the one hand a well-defined domain model describing all relevant domain items and on the other hand an exhaustive user model. For both of these we already defined specific requirements in the two previous subsections.

As shown, adaptation can be applied in many parts of the content retrieval and distribution pipeline. However, in this dissertation we mainly focus on how we can adapt information retrieval and access to heterogeneous data sources. Therefore, also the requirements for the adaptation process are targeted at this subfield. As a consequence, the adaptation process is activated whenever the user wants to obtain information about domain items. In this respect, we require two types of user-adapted data access. Firstly, the user should be able to actively search for items, with the adaptation process involved to help the user in finding the best matches available. Secondly, we require a recommendation facility which can, without any active involvement of the user, recommend the set of items which fit the user best at any particular point in time, again by taking his or her user model into account.

3.3 Choosing the data model

In the previous chapter, we saw an overview of the most important metadata specifications for audiovisual content currently available. Further, a selection was made including five metadata specifications most suited to support interactive and personalized television applications. Here, we want to investigate whether any of these five standards qualifies as foundation for our domain model, taking the previously defined requirements into account. While doing so, we have to keep in mind that every one of these specifications was originally built to solve a specific problem possibly different from ours.

In this section, we show for each of the five remaining specifications a metadata example by which we can discuss every standard’s advantages and disadvantages with respect to our requirements.

3.3.1 Escort 2.4

As explained in Chapter 2, Escort 2.4 is a standard developed by the European Broadcasting Union (EBU) and the predecessor of Escort 2007. However, since Escort 2007 was not released yet at the start of this dissertation we discuss the version available in those days, namely Escort 2.4. As previously explained, Escort 2.4 annotates television and radio programs by classifying them by means of a number of elementary dimensions. In each dimension, only one code can be given to describe a piece of content, and the digits of any such code together determines its position in the hierarchy. The first digit stands for the main class, the second refers to a subclass of the first class, the third is a subclass of the second, etc. In Table 3.1 we show an annotation example in the escort 2.4 structure. The numbers in these dimensions map as follows: format:2.2.1 => “Performed drama”, format:2.1.3 => “Commented Event”, content:3.2.1.5 => ‘Soap’, content:3.1.4 => ‘Sports’, target:4.1 => “General Audience”, origin:5.2 => “Concert hall/Theater”, origin:5.4 => ‘Videogram’, cycle:3.4.3 => “long-running serial” and cycle:3.1 => “Single production”. Together, the selected terms from these dimensions give a good impression of the type of program we are dealing with.

Program annotation via multiple dimension classifications is a logical approach and was adopted by many others afterwards (e.g. MPEG-7, TV-Anytime, etc.). However, the Escort standard solely depends on these classifications, limiting its descriptive power. It is for example impossible to assign a synopsis, production year, credit list with actors, broadcast channel, etc. to a program, because values of such properties are impossible to capture by means of classifications. Moreover,
3.3. CHOOSING THE DATA MODEL

<table>
<thead>
<tr>
<th>Title</th>
<th>Format</th>
<th>Content</th>
<th>Target</th>
<th>Origin</th>
<th>Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Eastenders</td>
<td>2.2.1</td>
<td>3.2.1.5</td>
<td>4.1</td>
<td>5.1</td>
<td>3.4.3</td>
</tr>
<tr>
<td>Hamlet (from theatre)</td>
<td>2.2.1</td>
<td>3.2.2.1</td>
<td>4.1</td>
<td>5.2</td>
<td>3.1</td>
</tr>
<tr>
<td>Soccer match</td>
<td>2.1.3</td>
<td>3.1.4</td>
<td>4.1</td>
<td>5.4</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Table 3.1: Example of an Escort 2.4 program annotation

every kind of free textual information not captured in an existing classification cannot be used to annotate a program. Also, referencing other audiovisual resources like e.g. related programs, pictures, trailers, etc. cannot be modeled with Escort 2.4. After all, Escort 2007 was partially conceived because of some of these limitations. All things considered, we have to conclude that Escort 2.4 is not a good candidate to fulfill our domain model requirements.

3.3.2 XML-TV

In Chapter 2 we saw that XML-TV is a standard developed by a community of people who shared the need for a system that could extract available online program listings. They created a model to contain this data (defined by the XML-TV DTD as shown in Appendix A.1.1) which is now used by various Electronic Program Guides like for example MythTV, epgStream.net, etc. In Figure 3.1 we see an example of an XML-TV program description. The first thing that stands out when looking at this example is the fact that the standard is built very pragmatically. It contains exactly those properties it needs to tell what this program is about, and nothing more. It contains a title, channel, genre (category), synopsis (desc), start time, country of origination and a list of credits with specific roles attached.

```xml
<tv generator-info-name="myListingsGenerator">
  <channel id="ned1.nl">
    <display-name lang="de">Nederland 1</display-name>
  </channel>
  <channel> ... </channel>
  ...
  <programme start="20090616225130" channel="ned1.nl">
    <title lang="nl">Pauw & Witteman</title>
    <desc lang="nl">
      Peter Paul de Vries over het afgelopen crisisjaar,...
    </desc>
    <category lang="nl">Talkshow</category>
    <credits>
      <presenter>Jeroen Pauw</presenter>
      <presenter>Paul Witteman</presenter>
    </credits>
    <date>20090616</date>
    <country>NL</country>
    <video><aspect>16:9</aspect></video>
    <star-rating><value>3/3</value></star-rating>
  </programme>
  ...
</tv>
```

Figure 3.1: Example of an XML-TV Metadata description

---

2.http://epgstream.net/
However, also here we see some shortcomings. To start, this specification is built by a community of amateurs in 2002, and therefore not really ready for the future demands. There is for example no way to reference related content, programs, pictures, video, etc., although you can add a URL reference to the Web site of a particular program. There is also no way to describe more fine-grained parts of a program like for example specific scenes or segments. The genre (category) approach is also very rudimentary. The category property just contains whichever genre string is found on the listings' source. Therefore there is no structure or guideline whatsoever concerning how different genres relate to each other (what you for example do have in Escort 2.4 where genres are classified in hierarchies). Again, also this specification cannot be reconciled with the requirements previously determined.

3.3.3 LSCOM
The consortium of the Large-Scale Concept Ontology for Multimedia (LSCOM) [155] consortium endeavors for the creation of a large taxonomy for describing broadcast news video. In other words, LSCOM provides a taxonomy which you can use to, either manually or preferably automatically, tag television programs and in particular, news programs. Moreover, LSCOM was originally created to serve as a tool for semantic analysis of multimedia content. Although LSCOM is a very well structured means to classify programs, it can not be considered as an exhaustive audiovisual data model, making it unsuited for our endeavor.

3.3.4 TV-Anytime
TV-Anytime, as described in Chapter 2, is a content description framework which allows the consumer to find, navigate and manage content from a variety of internal and external sources including, for example, enhanced broadcast, interactive TV, IP data and local storage [92]. TV-Anytime was built to make a television program description as richly as possible while maintaining a good perspective over potential future developments. The TV-Anytime specification contains constructions to model almost any aspect of a TV program as accurate as possible:

- Program descriptions: Including title, synopsis, genres, production location, keywords, sign-, caption- and spoken language, credit lists, award lists, etc.
- Program locations: Modeling all types of media and sources governing for example broadcast streams, Video-On-Demand (VOD), pay-tv, etc.
- Classifications Schemes: TV-Anytime includes a variety of different classification schemes (mostly adopted from standards like Escort 2.4, MPEG-7, etc.) to model genres, formats, intended audiences, content origination, etc.
- Segmentation annotation: Every program segment can have its own distinct metadata describing that particular segment.
- Packaging: Through the concept of a Package, various links and connections can be made between different programs or resources like pictures, audio, video, text, etc.
- Instance description: Different instances of a specific program sometimes require additional metadata including e.g. content location, delivery parameters, etc.

While TV-Anytime is one of the most complete structures to model television programs and is widely supported by big players in the content delivery chain, there are also some disadvantages. Since TV-Anytime was developed by a very large consortium, sometimes compromises were made in terms of structure and compatibility. This mainly resulted in a sometimes illogical structuring of concepts and/or almost unreadable files. Also, the structure of the classification hierarchies is sometimes at least questionable. To give a simple example, in the content classification hierarchy the term for ‘Religion’ (which is a not that extremely important genre in television programs) has
more than 35 different narrower terms (kinds of religions), while on the other hand an important
term like ‘Comedy’ (which is a very diverse and often occurring genre) only has six narrower terms.
On the positive side, TV-Anytime does support the addition of pictures, video, related material
and any other type of future media.

TV-Anytime uses its own way to uniquely identify resources. It does so via CRIDs (Content
Reference Identifier). However, while it remains the vision of the TV-Anytime forum to make
CRIDs as universal as a URL, this is still not the case. Therefore CRIDs are not that universal
yet, and not much servers around understand the “crid://” prefix. However, in 2005 the Internet
Engineering Task Force (IETF) published a request for comments specifying the use of the CRID
over the Web. This would allow consumer devices to hook up to content provider servers, much
like current browsers look up Web servers, enabling the request for content descriptions by means
of a CRID.

```xml
<ProgramGuideML>
  <ProgramTable>
    <ProgramItem order="1">
      <ProgramContent programContentId="nhk-tv-general-pc">
        <Body newsml_formalname="Newspaper-A_Long">
          <BodyContent xml:lang="en-GB">
            <P xml:lang="en-GB">12:00 News,:20(S)Lunchtime</P>
          </BodyContent>
        </Body>
        <SubProgram order="1">
          <SubProgramInformation>
            <tva.ProgramDescription> #TV-Anytime program and location data</tva.ProgramDescription>
          </SubProgramInformation>
          <SubProgramContent programContentId="nhk-tv-general-spc">
            <Body newsml_formalname="Web">
              <BodyContent xml:lang="en-GB"/>
            </Body>
            <Body newsml_formalname="CellularPhone">
              <Modification>decision</Modification>
              <BodyContent><P xml:lang="en-GB">12:00 News</P></BodyContent>
            </Body>
          </SubProgramContent>
        </SubProgram>
      </ProgramContent>
    </ProgramItem>
  </ProgramTable>
</ProgramGuideML>
```

Figure 3.2: Example of a ProgramGuideML Metadata description

### 3.3.5 ProgramGuideML

In Chapter 2 we saw that ProgramGuideML works as an extra layer on top of TV-Anytime.
TV-Anytime is used to model the television programs, while ProgramGuideML adds extra informa-
tion necessary for its specific goal: a format to store radio and television program information,
supporting a custom layout for news related facilities (as shown in Figure 3.3). In Figure 3.2 we
see some illustrative ProgramGuideML code (do note that due to space constraints some con-
structs are slightly shortened). In the code of Figure 3.2 we see the extra metadata added by
ProgramGuideML. The TV-Anytime constructs have been removed, and replaced by a line start-
ing with ‘#’ to indicate the location. ProgramGuideML encloses TV-Anytime blocks between
```xml
<tva.ProgramDescription> tags, as shown in the figure. While most of the metadata is modeled
```
in TV-Anytime, some extra information is added. In the ProgramContent block we see that they added, via a system of subprograms, extra information relevant for the client system receiving it. We see for example metadata statements either valid when shown on the Web or when the target client is a mobile phone.

Ultimately, the main focus of ProgramGuideML is to store radio and television program information, modeled in TV-Anytime, but tuned to be published in program guides. Via its approach, it benefits from all advantages offered by TV-Anytime, however the additions made do not really add anything useful with respect to our specific goal and requirements. Therefore, we discard ProgramGuideML as a potential solution since TV-Anytime itself would then be preferred.

### 3.3.6 Conclusions

In this section, we compared the five remaining specification candidates. Here, we found that two are not really describing television programs but rather provide term hierarchies to classify them (Escort 2.4 and LSCOM). Further, we saw XML-TV which is a community developed schema which is nice to feed an EPG but misses the profundity to describe television programs in all of its aspects. ProgramGuideML is a specification built on top of TV-Anytime but with a more specific focus, namely for news agencies to publish EPG data. However, the extra metadata maintained by ProgramGuideML is of limited use as it focusses strongly on this publishing.

The last remaining candidate is TV-Anytime. TV-Anytime is a schema which was specifically made by a professional consortium of television related companies for next generation television systems. Despite of a few minor issues, TV-Anytime contains constructs to express almost every possible feature of a television program in considerable depth. Further, it provides the necessary structures to richly and completely describe any given television program. Moreover, it has been setup with the idea that you do not always need to reinvent the wheel, as it makes extensive use of already existing schemes like MPEG-7 and Escort hierarchies. Concerning these hierarchies, TV-Anytime includes a large number of them to classify any given program from many different perspectives.

### 3.4 TV-Anytime

In Chapter 2 we previously discussed the general overview of the TV-Anytime specification. However, now that is has further become clear that TV-Anytime is the best candidate for the deployment of a TV domain model, considering our requirements, we provide a more technical and “under the hood” overview of how TV-Anytime is constructed and operates.
3.4. TV-ANYTIME

3.4.1 Introduction

TV-Anytime is a full and synchronized set of XML specifications established by the TV-Anytime forum built to enable search, selection, acquisition and rightful use of content from both broadcast and online services [238]. It basically consists out of two main parts usually referred to as TV-Anytime Phase I and Phase II. Phase I basically provides the metadata constructs to model and describe any specific television program, while Phase II describes possible interrelations to cluster different resources like programs, audio files, video files, texts, etc. in the form of packages.

As mentioned in Chapter 2, the CRID (Content Reference IDentifier) is the cornerstone in any TV-Anytime program description. The CRID, described in ETSI TS 102 822-4, provides a unique and unambiguous identification of every entity described following an RFC standard [84]. Such an entity can be any resource, or in some cases also a group of resources. A CRID can for example reference a group of resources sharing a particular trait and clustered as a package. The syntax of a CRID looks like:

CRID://<authority>/<data>

Here the <authority> is a registered Internet domain name while <data> is a free format string which is URI (Uniform Resource Identifier) compliant and meaningful. This portion of the field is case insensitive, however, it is recommended that all characters not within the range of characters allowed in a URI must be encoded into UTF-8 and included in the URI as a sequence of escaped octets.

With such a CRID, which always uniquely identifies a resource, we can retrieve the resource’s metadata and obtain further references to other resources. However, with a CRID, which behaves not so different from a URL, there is a need for a mechanism which resolves the CRID and returns the correct response, just like a Web server returns an HTML file for a URL request. To this end, the TV-Anytime specification describes the concept of a CRID Authority (CA) which main purpose is to resolve CRIDs. When resolving a CRID at the CA, the CA returns either a set of locators or a set of CRIDs. Each locator contains the actual location of the content element(s) which belong to the referenced program. When a CRID resolves into multiple other CRIDs, they reference more specific subparts of the original CRID. In Figure 3.4 we see the resolving process when a CRID is sent to the CA. Because one CRID can resolve to multiple locators, it is possible that one resource resides in multiple locations. Those different locations can both contain exact the same resource or different incarnations of that resource (e.g. with lower/higher resolution, different sound quality, etc.). Therefore, in certain situations the most appropriate content location can be chosen in terms of availability, quality, connection speed, etc., but also on criteria like user preferences, terminal capabilities (e.g. device capabilities), user input, etc.

Further, besides resolving CRIDs, we can also obtain the associated metadata of the resource uniquely identified by this CRID. The TV-Anytime Metadata Service (MS) is the entity described by the TV-Anytime specification to be responsible for this task. For every existing CRID, the MS can provide a metadata description, given that whomever initially created the resource (referenced by this CRID), registered a correct TV-Anytime metadata description at the MS. To further clarify the dynamics and communications between these services, we provide here a small scenario to further illustrate it in combination with a PVR (Personal Video Recorder):

![Figure 3.4: CRID resolving process](image-url)
CHAPTER 3. REQUIREMENTS

By means of a CRID, every TV program is uniquely identified. At the CA, the CRID of the program $P$ resolves for example into a DVB-locator which indicates the channel and time of $P$’s broadcast. However, when a PVR is instructed to record $P$, it is not the DVB-locator that is used to program the event, but rather the CRID itself which always remains unique. The advantage is that when the locator changes, the PVR only notices this when it resolves the CRID at the CA. This allows the publisher to change for example the date, time, length of the program at any time, while the PVR (which has already scheduled to record $P$), still records the program correctly. At time of broadcast, the PVR requests the latest DVB-locators, indicating the location of the effective content, and starts recording. At any time, the metadata of the program can be requested at the MS to show the user what this program is about.

3.4.2 TV-Anytime Phase I

The TV-Anytime Phase I specification is a very extensive metadata schema which can describe all kinds of television, and television related, content. As previously noted, fundamental requirement for searching and filtering. As described in the TV-Anytime specification, Phase I consists of four main metadata types among which we find:

- **Content description metadata**: General information about a piece of content that does not change regardless of how the content is published or broadcast. It includes information such as the content’s title, textual description and genre(s). Typically, the content creator assigns content description metadata before publication.

- **Instance description metadata**: Describes a particular instance of a piece of content, including information such as the content location, usage rules (pay-per-view, etc.) and delivery parameters (e.g. video format). Instance description metadata is assigned by the content provider as a part of the publication of content. During the search and selection process, a consumer may use both general content and instance descriptions. Instance description metadata is useful in cases where there are meaningful differences between instances of the same content (that is, instances of content that share the same CRID). Instance description metadata is linked to a particular event-related instance of content.

- **Consumer metadata**: Includes usage history data (logging data), annotation metadata and user preferences. Usage history information is gathered over extended periods of time by e.g. a PVR (Personal Video Recorder). The collected usage history provides a list of the actions carried out by the user during an observation period, which can subsequently be used by automatic analysis methods to generate user preferences.

- **Segmentation metadata**: allows to define, access and manipulate temporal intervals (i.e. segments) within an AV stream. By associating metadata with segments and segment groups, it is possible to restructure and re-purpose an input AV stream to generate alternative consumption and navigation modes. Such modes could include, for example, a summary of the content with highlights, or a set of bookmarks that point to “topic headings” within the stream. Such metadata can be provided by service providers or broadcasters as a value-added feature and/or generated by viewers themselves.

Modeling these four main metadata types, phase I consists of a number of self-contained documents. The most important of these documents, contains the main TV-Anytime root called TVAMain, which contains the references to all relevant metadata structures describing television programs. The structure of such a TVAMain document can be seen in Figure 3.5. Next to some general metadata properties like MetadataOriginationInformationTable (where does the metadata come from), ClassificationSchemeTable (inclusion of the classification hierarchies referenced in the metadata), CopyrightNotice (who has the copyright on this metadata) and UserDescription (a description of the user can be sent if he or she is known), the TVAMain root exists of eight tables (bundled under ProgramDescription) of which each describes a specific aspect of the program’s
metadata. Therefore, every program can have an (optional) entry in each of these tables. In other
words, every table contains metadata entries of a whole collection of programs (e.g. all programs
broadcasted that day) identified by their CRIDs. Taking from these tables all the metadata entries
of one specific program $P$ together, defines the complete known metadata description of $P$. These
eight tables include:

- **ProgramInformationTable**: Includes all metadata fields providing information about $P$ re-
gardless of how the content is published or broadcast. E.g. title, synopsis, credits, etc.

- **GroupInformationTable**: Describes a list of groups to which a program $P$ might belong. A
  program can belong to several groups and groups themselves can be nested. E.g. an episode
  of ‘Friends’ can belong to both the group ‘Friends’ (modeling the series) and “Friends Season
  1” (modeling the first season). In which case, the first season itself also becomes a member
  of the series as a whole.

- **ProgramLocationTable**: States either where $P$ can be found (in case of Video-On-Demand)
  or when $P$ will be broadcast (in case of a regular broadcast).

- **ServiceInformationTable**: Describes the service responsible for making the program available
  for the audience. Service providers can be online services, VOD suppliers, broadcasters, etc.

- **CreditsInformationTable**: This table contains an overview of all persons (e.g. actors, direc-
tors, presenters, guests, etc.) referenced in programs described in any of the other metadata
  tables like e.g. the ‘ProgramInformationTable’.

- **ProgramReviewTable**: Contains a list of program reviews.

- **SegmentInformationTable**: Contains additional metadata for specific program segments, like
  for example the exact segment in a soccer match where the goal gets scored.

- **PurchaseInformationTable**: In case a program is not available for free, this table contains
  the information about how and where it can be purchased.

From this list, three tables are especially relevant since they provide the metadata describing the
‘what’ and ‘where’ of the program: the ‘ProgramInformationTable’, ‘ProgramLocationTable’ and
‘ServiceInformationTable’. In Figure 3.6 we see an example of a TV-Anytime program description
with a particular emphasis on these three tables. For further details of the metadata specification
we would like to refer to the *ETSI TS 102 822-3-1* specification document.
CHAPTER 3. REQUIREMENTS

<table>
<thead>
<tr>
<th>Program Information Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong>: News</td>
</tr>
<tr>
<td><strong>Synopsis</strong>: News of 12 o'clock</td>
</tr>
<tr>
<td><strong>Keyword</strong>: News</td>
</tr>
<tr>
<td><strong>Genre</strong>: Daily news</td>
</tr>
<tr>
<td><strong>Parental Guidance</strong>: 0</td>
</tr>
<tr>
<td><strong>Language</strong>: en</td>
</tr>
<tr>
<td><strong>Depicted Coordinates</strong>: BBC Studio</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Audio Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Channels</strong>: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Video Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AspectRatio</strong>: 4:3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program Location Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Broadcast Event</strong>:</td>
</tr>
<tr>
<td><strong>StartTime</strong>: 2009-04-13T12:00:00</td>
</tr>
<tr>
<td><strong>EndTime</strong>: 2009-04-13T12:20:00</td>
</tr>
<tr>
<td><strong>Duration</strong>: P0Y0M0DT0H20M</td>
</tr>
<tr>
<td><strong>Live</strong>: true</td>
</tr>
<tr>
<td><strong>Repeat</strong>: false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service Information Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service</strong>: BBC World</td>
</tr>
<tr>
<td><strong>URL</strong>: <a href="http://www.bbcworldnews.com">http://www.bbcworldnews.com</a></td>
</tr>
</tbody>
</table>

Figure 3.6: Example of a TV-Anytime Metadata description
To maintain some control and structure in the annotation process of television programs, TV-Anytime has adopted and extended the principle of program classification via controlled hierarchies as seen in MPEG-7 and Escort 2.4. Through these generally acknowledged classifications, metadata creators can classify their programs for a broader audience, improving interoperability between different parties. The genre field in TV-Anytime for example is, among others, used to refer to terms of these hierarchies. TV-Anytime contains in total seventeen hierarchies\(^3\), each describing a particular facet of a program. However, MPEG-7 hierarchies can still be referenced from within TV-Anytime metadata and moreover, new or custom hierarchies can be defined. Standardly TV-Anytime includes:

- **ActionTypeCS**: Types of user actions which can be monitored to analyze viewing behavior. It includes actions like e.g. ‘PlayStream’, ‘Record’, ‘Preview’, ‘Skipforward’, ‘Mute’, etc.

- **HowRelatedCS**: A series of definitions for possible relations between different resources like programs, video, advertisements, Web sites, etc. These relations therefore include e.g. ‘Has-Trailer’, ‘Parent’, ‘HasMakingOf’, ‘Derived’, etc.

- **TVARoleCS**: Key cast roles including e.g. “Costume designer”, “Director of photography”, “News reader”, “Interviewed guest”, etc.

- **IntentionCS**: The broadcaster’s primary apparent intention in transmitting the program. Such intentions can be to e.g. ‘Entertain’, ‘Inform’, ‘Educate’, ‘Advertise’, etc.

- **FormatCS**: This dimension is used to classify programs to their formal structure. In other words, how does the program look, regardless of the subject with which the program is dealing. Examples are e.g. ‘Magazine’, ‘Event’, ‘Documentary’, “Talk show”, etc.

- **ContentCS**: This classification is used to classify programs according to their content or subject. Unlike in the case of the format dimension, it is essential to hear/see the program before one can elicit the subject. Examples are e.g. ‘News’, ‘Soccer’, ‘Action’, ‘Comedy’, etc.

- **ContentCommercialCS**: A hierarchy with genres specifically focussing on the commercial products. It can be seen as a commercial extension of the ContentCS classification. It includes for example “Metal ores”, ‘Water’, “Tobacco products”, ‘Chemicals’, etc.

- **OriginationCS**: The original distribution method or platform for the content. Including e.g. ‘Cinema’, ‘Live’, “Made by the consumer”, “Made in the studio”, etc.

- **IntendedAudienceCS**: Describing the audience this program was intended for, defined by age, cultural/ethnic background, profession etc. Examples are e.g. ‘Children’, “Immigrant groups”, ‘National’, ‘Single’, ‘Retired’, etc.

- **ContentAlertCS**: Terms indicating potential warnings concerning the content of the program. These terms are mainly used for parental control on television programs. Among them we find for example ‘Sex’, ‘Nudity’, ‘Violence’, ‘Language’, ‘Discrimination’, etc.

- **MediaTypeCS**: The type of the program in terms of its media including e.g. “Audio only”, ‘Multimedia’, ‘Application’, ‘Data’, etc.

- **AtmosphereCS**: A classification which contains terms conveying the psychological, emotional and subjective assessment of a program. Includes e.g. ‘Chilling’, ‘Compelling’, ‘Fun’, ‘Happy’, ‘Inspirational’, etc.

- **AudioPurposeCS**: Classifies the purpose of the audio in the program. Examples are for example “Supplemental commentary”, “Audio description for the visually impaired”, etc.

\(^3\)With every new version of the ETSI TS 102 822-3-1 specification new hierarchies are introduced. Currently, in version 1.4.1 there are seventeen TV-Anytime hierarchies listed.
• PurchaseTypeCS: Contains three basic types for the purchase of content: “Play forever”, “Play for period”, “Play counts” (a limited number of plays).

• UnitTypeCS: Defines units of time to further quantify the validity of a purchase. The units include ‘Days’, ‘Months’ or ‘Years’, etc. for a period of time and ‘Plays’ to quantify a fixed number of times a person can play the content.

• DerivationReasonCS: Sometimes, a new version of a program is derived from the original. This can have different reasons like e.g. ‘Violence’, ‘Duration’, “Special cut”, etc.

• CaptionCodingFormatCS: There exist different formats to model the captions of a program. In this hierarchy the terms refer to the type of caption including for example “DVB Subtitles”, “WST Subtitles”, etc.

Obviously there is a difference in detail in these classifications. Some are very rigorous descriptions of a domain (e.g. Format classification) and some are nothing more than a mere flat list of terms (e.g. TV-Anytime Role Classification). Also the content genre classification is a good example of a fine grained taxonomy structure, going from general concepts like ‘Fiction’ or “Non-fiction” down to specific categories in the leaf nodes of the structure (typically well known genres like ‘Comedy’, ‘Drama’, “Daily news”, “Weather forecast”, etc.). In Figure 3.7 we see a small part of the TV-Anytime content classification hierarchy. Every term (here a genre) has an id (termID) which exemplifies its depth in the tree and makes it easy to see which other term is the parent (e.g. term 3.1 is the parent of term 3.1.1). For example, the term “Sport news” is a specialization of the term ‘News’ which in turn is narrower than the term “Non-fiction/Information”.

```xml
<ClassificationScheme uri="urn:tva:metadata:cs:ContentCS:2005">
  <Term termID="3.1">
    <Name>NON-FICTION/INFORMATION</Name>
    <Term termID="3.1.1">
      <Name>News</Name>
      <Definition>Time-sensitive information</Definition>
      <Term termID="3.1.1.1">
        <Name>Daily News</Name>
        <Definition>Regular program carrying breaking and current news stories</Definition>
      </Term>
    </Term>
    ...  
    <Term termID="3.1.1.9">
      <Name>Sport News</Name>
      <Definition>Sport news</Definition>
    </Term>
  </Term>
  ...  
</ClassificationScheme>
```

Figure 3.7: Example of a part of the TV-Anytime content classification hierarchy

### 3.4.3 TV-Anytime Phase II

Connecting different pieces of content is a logical evolution in content provisioning. Whether this is done through the similarity of certain attributes of those items, via shared interests in a community (e.g. Amazon’s “Maybe you are also interested in...”) or via clustering (e.g. connecting all episodes of Friends), it all serves the need for efficient navigation through structured information. Also TV-Anytime accommodates clustering of programs via its *packaging* concept, described in Phase II of
3.4. TV-ANYTIME

the specification. This technology, enables the combination of different types of content items such as games, applications and interstitial content with audio, video, still images and text, to create an extended user experience. A package consists of a hierarchical collection of content items which share some traits and are intended to be consumed together providing a more profound consumer experience. For example, one could have an audiovisual French language course with an accompanying word game, situational movie clips, spoken audio, etc. to support people in learning the French language.

As described in the *ETSI TS 102 822-3-3* specification, a package is defined as a hierarchical collection of *Items*, where an item is a container providing access to one or more consumable entities, logically belonging together within this item. An item therefore represents a kind of multimedia cluster as a unit of consumption. In addition, an item itself can also contain other items which further refine the parent item. A *Component* on the other hand, is a single acquirable entity of content pointing to one unique resource like for example, a specific MPEG video/audio file, a Web site, a piece of text, etc. Consequently, every item can consist out of a new set of items, a set of components or a combination of both.

The Phase II TV-Anytime Metadata Schema is a backwards-compatible extension of Phase I. It extends Phase I data types for content description and user description and makes use of imported data types from MPEG-21 to enable new areas of functionality. It also extends the TV-Anytime root document type, TVAMain, with a *PackageTable* to model the package structure. The data model of a package adopts the multi-level structure as seen in the MPEG-21 Digital Item Declaration Language (DID) [175], i.e. the container-item-component structure, along with some refinements and extensions. In the following code snippet we see a rough example of this container-item-component structure as it appears in the *PackageTable*:

```xml
<tva2:PackageTable crid="crid://www.ca.org/ex1">
  <tva2:Item>
    <tva2:Component>
      <tva2:Resource mimetype="video/mpeg" url="" />
    </tva2:Component>
    ...<tva2:Component>
      <tva2:Resource mimetype="image/jpeg" url="" />
    </tva2:Component>
  </tva2:Item>
  <tva2:Item>...</tva2:Item>
  ...
</tva2:PackageTable>
```

To illustrate a package with a practical example, we adopted the BBC’s “French course”⁴. This course consists of a number of lessons describing for example “Getting around in France”, “Organizing a day trip”, etc. which are each subdivided in a number of situations (like “Short break in Paris”, ‘Chatting’, “Food and drinks”, etc.). Each of this situations is then further illustrated by means of small everyday things like “Taking a taxi”, “Talking about the family”, “Shopping for food”, etc. as shown in Figure 3.8.

When we turn this French course into a package, first we need to define proper CRIDs for every node and content element within the package, and define the structure by means of items, components and locators. The French course package is divided in two lessons which each are further subdivided in three parts, following the structure depicted in Figure 3.8. In Figure 3.9 we see the structure of the package, although only the first part of lesson 1, “Short break in Paris”, is further elaborated. Each of the three specific tasks in this part is modeled as a subitem under the central item, where each further has a set of components. Each component then points to

⁴http://www.bbc.co.uk/languages/french/lj/menu.shtml
## CHAPTER 3. REQUIREMENTS

**Figure 3.8: French course lessons overview**

<table>
<thead>
<tr>
<th>Lesson 1: Short break in Paris</th>
<th>Lesson 2: Country holiday</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Topics</strong></td>
<td><strong>Topics</strong></td>
</tr>
<tr>
<td>Taking a taxi</td>
<td>Buying clothes</td>
</tr>
<tr>
<td>Asking for directions</td>
<td>Understanding instructions</td>
</tr>
<tr>
<td>Getting a snack</td>
<td>Talking about the weather</td>
</tr>
<tr>
<td>Chating</td>
<td>Renting item</td>
</tr>
<tr>
<td><strong>Topics</strong></td>
<td><strong>Topics</strong></td>
</tr>
<tr>
<td>Ordering at a restaurant</td>
<td>Asking friends round</td>
</tr>
<tr>
<td>Shopping for food</td>
<td>Arranging to go out</td>
</tr>
<tr>
<td>Asking about the food</td>
<td>Taking the train</td>
</tr>
<tr>
<td>Food and drink</td>
<td>Booking a room</td>
</tr>
<tr>
<td><strong>Topics</strong></td>
<td><strong>Topics</strong></td>
</tr>
<tr>
<td>Chating</td>
<td>Renting item</td>
</tr>
<tr>
<td><strong>Topics</strong></td>
<td><strong>Topics</strong></td>
</tr>
<tr>
<td>All the chemists'</td>
<td>On the road</td>
</tr>
<tr>
<td>On the road</td>
<td>Lost property</td>
</tr>
</tbody>
</table>

**Figure 3.9: French course package structure**

- ![Package: “French Course”](image)
- ![Item: “Lesson 1”](image)
- ![Item: “Lesson 2”](image)
- ![Item: “Chatting”](image)
- ![Item: “Short break in Paris”](image)
- ![Item: “Food and drinks”](image)
- ![Item: “Taking a Taxi”](image)
- ![Item: “Asking for directions”](image)
- ![Item: “Getting a snack”](image)
- ![Component: “The taxi”](image)
- ![Component: “Directions”](image)
- ![Component: “Snacks”](image)
- ![Locator: “The taxi”](image)
- ![Locator: “Directions”](image)
- ![Locator: “Snacks”](image)
consumable content elements by means of a set of locators. The item “Taking a Taxi” for example, resolves into two components where the first represents a video and the second a piece of text. By means of the component’s unique CRID, locators can be obtained by resolving it in the CRID Authority. The video component for example resolves into two locators, both pointing to the same video. However, the first locator points to a DVB stream while the second references an HTTP location on the Web. Because content elements are not stored in the package itself, but referenced using a locator pointing to the resource, the effective content is not fixed and can be changed or updated at any time. In Figure 3.9 we further see that besides DVB and HTTP locations some parts can also be distributed via a disc (e.g. the audio of the component ‘Directions’).

When the CRID of the “Short break to Paris” item is sent to the CRID Authority (CA), we receive a XML document containing the structure as seen in Figure 3.9 following the process depicted in Figure 3.4. The following XML code snippet shows the response of the CA when asked to resolve both the CRID of the item “Short break to Paris” and the component “The Taxi” respectively:

```xml
<Result CRID="crid://frenchcourse/lesson1/item/shortbreak"
  status="resolved" complete="true" acquire="all">
  <CRIDResult>
    <Crid>crid://frenchcourse/lesson1/item/taxi</Crid>
    <Crid>crid://frenchcourse/lesson1/item/askForDirection</Crid>
    <Crid>crid://frenchcourse/lesson1/item/gettingASnack</Crid>
  </CRIDResult>
</Result>

<Result CRID="crid://frenchcourse/lesson1/component/taxi"
  status="resolved" complete="true" acquire="all">
  <LocationsResult>
    <Locator>http://www.bbc.co.uk/french/1j/taxi/m1.avi</Locator>
    <Locator>dvb://4e.3f4;4f@2009-03-27T18:00:00.00?:00</Locator>
  </LocationsResult>
</Result>
```

Because of the dynamic nature of the packaging concept, any package can easily be modified or extended with new parts. A new chapter could for example easily be implemented by adding an extra CRID reference accompanied by more items, components and content locators.

### 3.5 The Semantic Web

So far we discussed the TV-Anytime metadata specification, specialized in describing television programs in various aspects. We also saw that TV-Anytime, which can be considered to be a powerful and expressive television program metadata specification, also has some notable disadvantages. Moreover, the biggest and most noticeable drawbacks are not so much due to TV-Anytime itself, but rather because XML was chosen as representational language. As a short introduction, an XML document can be seen as a tree of nested sets of tags. Each of those tags can have a set of attributes with an associated value. Standardly, there are no restrictions or grammars to control the structure of the document. However, if necessary, the XML description can be restricted by specifying an allowed structure in either a Document Type Definition (DTD) or an, more recently introduced, XML Schema Document (XSD) file.

#### 3.5.1 XML’s Shortcomings

To illustrate, in this example we introduce program $P$. The descriptive TV-Anytime metadata of $P$ is shown in its original XML format, depicted in Figure 3.10. $P$ represents the movie “The Alzheimer Affair” directed by “Erik Van Looy”. The movie was produced in 2004 and was filmed
in various locations among which “Antwerp, Belgium”. However, this is how we, as humans, read and understand this information. However, what would a computer, being the system which is supposed to do all of the metadata processing, ‘understand’ from this data? Besides seeing a set of objects (which are mostly not uniquely identified), where each has some properties $X$ with a value $Y$, probably not much. Moreover, from this XML a computer does not ‘understand’ that “Erik Van Looy” is actually a person and that “BE-VAN” comprehends a physical location on the planet. This is a bit unfortunate, because such knowledge can be of great value to understand what data we are dealing with. Moreover, it becomes even more essential when we for example combine it with user data. Hence, if a user tells the system that he likes this movie a lot, it is interesting to know why exactly. Does he like it because it partially plays in Belgium, because he just loves recent movies or does he simply likes all movies directed by “Erik Van Looy”. From a different perspective, if the system wants to recommend programs to a user, on which ground could it decide whether or not to recommend this movie? Maybe because the user likes movies based on novels of “Jef Geeraerts”, or maybe because the user just loves Belgium.

```xml
<tva:ProgramInformationTable>
  <tva:ProgramInformation programId="crid://movie.xy/mov15498">
    <tva:BasicDescription>
      <tva:Title type="main">The Alzheimer Affair</tva:Title>
      <tva:CreditsList>
        <tva:CreditsItem role="urn:mpeg7:cs:RoleCS:2001:DIRECTOR">
          <tva:PersonName>
            <mpeg7:GivenName>Erik Van Looy</mpeg7:GivenName>
          </tva:PersonName>
        </tva:CreditsItem>
        <tva:CreditsItem role="urn:mpeg7:cs:RoleCS:2001:AUTHOR">
          <tva:PersonName>
            <mpeg7:GivenName>Jef Geeraerts</mpeg7:GivenName>
          </tva:PersonName>
        </tva:CreditsItem>
      </tva:CreditsList>
      <tva:ProductionDate>2004-04-15T00:00:00</tva:ProductionDate>
      <tva:CreationCoordinates>
        <tva:CreationLocation>
          <mpeg7:regionCode>BE-VAN</mpeg7:regionCode>
        </tva:CreationLocation>
      </tva:CreationCoordinates>
    </tva:BasicDescription>
  </tva:ProgramInformation>
</tva:ProgramInformationTable>
```

Figure 3.10: XML structure of a TV-Anytime described program

Achieving such a level of ‘understanding’ is for a computer system unobtainable given the given XML representation. Values like “BE-VAN”, “Jef Geeraerts”, “2004-04-15T00:00:00”, etc. have no meaning at all, since they are just strings describing the value of a property. These values cannot be interpreted as potential concepts or refer to existing resources. Moreover, these strings are not even uniquely identifiable, making it impossible to for example reference them from outside. Furthermore, if we would by any chance retrieve metadata of the same movie from another source using slightly different naming conventions putting for example “Geeraerts, Jef” instead of “Jef Geeraerts”, currently, these are two different strings and therefore falsely seen as two different values for the same property. The same holds for the value “2004-04-15T00:00:00”. Without extra interpretation logic, this string does not tell anything or cannot be used to for example compare the production date of two different programs.
Looking back at the requirements we listed at the start of this chapter, this issue introduces some severe limitation. Firstly, it is impossible to add a statement to the user model indicating that the user likes the location “BE-VAN” or the person “Erik Van Looy”. We can only add such statements to objects which are uniquely identifiable like the program itself or any classification term (via the CRID and URN respectively). Secondly, filtering or searching through badly structured elements is complex or close to impossible without an extra interpretation layer. For example, a query searching for all movies produced in the last ten years will not find this movie unless additional logic interprets the string “2004-04-15T00:00:00”. Lastly, in the current XML representation it is impossible to discover new statements by reasoning over the available knowledge. Under reasoning we understand every logical deduction supported by the data model. E.g. if $A$ leads to $B$ via property $r$ and $B$ leads to $C$ via property $r$, and we know that $r$ is a transitive property, then we can reason that $A$ also leads to $C$ via property $r$. Reasoning would for example help with queries searching for all programs produced in Belgium. In combination with the knowledge that Antwerp is actually located within Belgium, the reasoner would conclude that “The Alzheimer Affair” is indeed produced in Belgium and in turn lead to its selection for this query. However, given the current XML representation and therefore lack of reasoning capabilities, this movie would not be selected.

Concluding, we see that the main shortcoming of XML is the fact that it just describes grammars via either DTDs or XML schemas. There is no way to recognize a semantic unit from a particular domain because XML aims at document structure and imposes no common interpretation of the data contained in the document, as previously stated in [75]. Further, they conclude that XML is useful for data interchange between applications that both know what the data is about, but not for situations where new communication partners are frequently added. Unfortunately, this is what we expect will happen all the time when new sources, people, applications, etc. constantly sprout on the Web and need to work together.

### 3.5.2 A Web with meaning

The shortage of structure and meaning described in the previous section was also noticed by the original founder of the Web, Tim Berners-Lee in 2001. He envisioned a new Web which would bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users [36]. The difference with the current Web structure is that it was designed to make digital information readable for human users, while the new version would allow computers (or machines) to interpret and ‘understand’ the meaning of information. A Web supporting integration and sharing of data across different applications and boundaries. This new Web, which is know as The Semantic Web, is not a separate Web but rather an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation [36].

The central *lingua franca* of the Semantic Web is the XML based *Resource Description Framework* or RDF [157]. Although XML, and its limitations, was one of the reasons to argue that we needed a more structured expression language, it still serves as the syntactical language to express Semantic Web building blocks.

**RDF**

The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web [157]. It is designed to represent information in a minimally constraining, flexible way. It can be used in isolated applications, where individually designed formats might be more direct and easily understood, but RDF’s generality offers greater value from sharing. The value of information thus increases as it becomes accessible to more applications across the entire Internet [144].

Conceptually, RDF is built around two central concepts, *Resources* and *Properties*. Resources are used to model all imaginable entities (e.g. a document, person, car, image, tree, etc.) and
properties, which have a name, are used to define attributes of those resources. Since such properties can connect different resources, RDF can be seen as an *Edge Labeled Directed Graph* which can be traversed from resource to resource. Moreover, every resource and property is identified by its unique Uniform Resource Identifier or URI, consisting of a namespace and a local name. The namespace pinpoints the responsible authority, while the local name identifies a specific object within the authority’s data structure. In contrast to an XML description of an object, every RDF object is an independent entity on its own.

Because of this structure, any RDF document can be considered a set of *triples* where each represents a resource-property-value relation. One such a triple can also be referred to as a *statement* or *fact* within the larger collection of an RDF document. Each part of a triple has a distinct name:

- **Subject**: The resource to which the property applies.
- **Predicate**: The property which relates the resource to a specific value.
- **Object**: The value of the property for this specific resource.

The object in the triple can either be a resource which in turn again can have its own properties or a *literal* describing a concrete value for the property. It is this simple but expressive structure of RDF which provides the means to introduce statements and semantics to otherwise meaningless information in any document.

**RDFS**

As we just saw, RDF is the general-purpose language for representing information in the Web. However, an RDF document itself does not define the rules to play by. It does not specify any structure or restrict for example which properties are allowed for which resources, what type of value a property can have, etc. For this purpose, we have *RDF Schema* or RDFS. RDFS itself is a vocabulary in RDF which specifies how to describe RDF vocabularies [48].

RDFS is a language to construct domain-specific schemata and vocabularies which in turn are intended as “cooking book” for RDF descriptions. If a RDFS vocabulary defines that a person can own a car, the RDF description which is an instance of the schema, can state that person ‘John’ owns a specific ‘Volvo’ car. In Figure 3.11 we see an example of a small RDFS schema flanked by a corresponding RDF instance. However, just like DTDs or XSDs in a XML document, the RDFS schema does not force or restrict the RDF description in any way.

![Figure 3.11: RDF Schema definition including an illustrative RDF instance](image)

In RDFS, the structure of a vocabulary is determined via classes and relations (properties) between these classes. Any such a property has a *rdfs:domain* (the class which has the property as a feature) and a *rdfs:range* (the class or literal which defines the data type of the property).
Other essential properties in the RDFS description are `rdfs:subClassOf` (to denote class specializations), `rdfs:subPropertyOf` (to denote property specializations) and `rdfs:label` (to label any resource).

One of the nice advantages of RDFS is that it supports distributed schemata. Every RDFS schema can use classes and properties of any other schema known to exist. The idea is that you do not want to reinvent the wheel and reuse previously defined semantics into your own model. In the most utopian case you would ultimately end up with one big universal RDFS schema describing the world, in which various parties contributed that part of the domain in which they are considered to be experts.

**OWL**

For some applications, the building blocks available in RDF(S) just do not suffice to model the domain appropriately. For example, some applications require cardinality constraints (e.g. a property can only occur a minimum and/or maximum number of times) or some properties require to be defined as transitive (e.g. a `brotherOf` property) or some need the ability to utter that two classes are disjoint or equivalent, etc.

The **Web Ontology Language** or OWL for short, is designed for use by applications that need to process the content of information instead of just presenting information to humans [161]. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDFS (which can only describe semantics for generalization-hierarchies of RDF properties and classes), and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL comes in three different flavors:

- **OWL Lite**: Supports the creation of classification hierarchies and limited constraints. For example, while it does support cardinality constraints, it only permits cardinality values of 0 or 1.
- **OWL DL (Description Logics)**: Supports the maximum expressiveness of OWL (includes all OWL language constructs) while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time).
- **OWL Full**: Supports maximum expressiveness and the syntactic freedom of RDF, however without computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals as well as an individual in its own right.

For each application one must decide which OWL flavor fits best and know that allowing more expressibility comes at a price. Every OWL (Lite, DL, Full) document is an RDF document, and every RDF document is an OWL Full document, but only some RDF documents will be legal OWL Lite or OWL DL documents. Both RDF, RDFS and OWL are W3C\(^5\) recommendations.

**Ontologies**

Following the guidelines of the Semantic Web, the RDFS and/or OWL language(s) can be used to describe any domain model. Such a description consisting of a set of classes, relations and (optionally) predefined restrictions defining the domain, can be referred to as an **Ontology**. Or like formulated by Gruber: “An ontology is an explicit specification of a conceptualization.” [111].

Throughout this thesis we will often refer to knowledge structures, which model particular domains, parts of domains or general knowledge, by means of different terms like e.g. ontology, **vocabulary**, taxonomy, thesaurus, etc. However, these terms often lie conceptually very close to each other, making it hard to differentiate them properly. Considering their relevance in this thesis, we provide some definitions here following the notions of Pidcock as explained in [189]. A **controlled vocabulary** is a list of terms that have been enumerated explicitly and controlled by a registration authority. In this dissertation for reasons of convenience, we just use the term ‘vocabulary’ to refer

\(^5\)http://www.w3.org/
to a controlled vocabulary. A taxonomy on the other hand is a collection of controlled vocabulary terms organized into a hierarchical structure. Hence, each term in a taxonomy is involved in one or more parent-child relationships to other terms in the taxonomy. Further, a thesaurus is a networked collection of controlled vocabulary terms. This means that a thesaurus uses associative relationships in addition to parent-child relationships. The term ‘ontology’ is considered to be more general and often used to refer to a variety of data structures including glossaries, data dictionaries, thesauri, taxonomies, schemas, data models, etc. Therefore, we can say that a formal ontology is a controlled vocabulary, expressed in an ontology representation language like for example RDFS or OWL.

For more information concerning these terms and their definitions we would like to refer to [189].

3.6 Conclusion

In this chapter we addressed research question 2 (What are the requirements for an approach providing user-adapted data retrieval?). We investigated requirements from a domain-independent perspective to discover those which are relevant for user-centered domains where user-adapted access could support the user in retrieving the right items in the right context. Therefore, we looked at requirements with a special emphasis on the data model, user model and adaptation strategy. Concerning the data model it is for example essential to use a specification which can easily be extended and enriched by for example the integration of external data or adding referrals between items or between items and other sorts of resources. For the user model on the other hand, we require a well-structured model which can comprehend all, both explicit and implicit, relevant user information in a context-aware manner. Lastly, in order to effectively provide such access to large heterogeneous sources, we show that the main requirement involves the availability of a domain model thoroughly describing the relevant domain items, and a profound user model keeping track of the user’s relevant characteristics, preferences, interests, etc.

Previously in Chapter 2, we gave an overview of all audiovisual metadata specifications which could prove helpful in describing television programs. Based on their specification, we could there already select five which became potential candidates to serve as basis for the domain model in our illustrative domain. In this chapter however, we could, after the formulation of the requirements, finally select TV-Anytime as the most suitable specification currently existing to describe television programs. It contains all necessary constructs to describe any given TV program, while making extensive use of already existing schemes like MPEG-7 and Escort classification hierarchies.

However, like most existing specifications, TV-Anytime’s representational language is XML. While XML has a long and proven legacy, it also contains some shortcomings which are particular important to us. Mainly the fact that XML does not uniquely identify any object within its body and the fact that it is impossible to infer semantics from the available statements, was cumbersome. However, by applying the Semantic Web principle which introduces unique identifiers for resource and semantics to allow reasoning over the available statements, we could counter XML’s weaknesses. Furthermore, modeling our data model in Semantic Web languages like RDF(S)/OWL helps in making the structure extendable, and allows for the enrichment by means of other available sources.
Chapter 4

An Approach to provide User-adapted Access to Heterogeneous data sources

As illustrated in the previous chapters, the Internet is a flourishing place encompassing unlimited amounts of information and hundreds of millions of people simultaneously participating in the experience. However, we also saw that through the current information explosion the process of finding the right bit of information becomes increasingly difficult. However, having a lot of data does not necessarily imply difficulties in finding one specific subset. Database engineers have already invested decades of research in keeping huge databases still very responsive and scalable. The main difference with the Internet however, is the lack of semantic structure. HTML pages are currently just blobs of text which are given both a navigational structure and a presentational style. However, they lack the descriptive structure which could include for example topic, author, publishing year, credibility, links to related sources, etc. which could be used by automated processes to ‘understand’ its contents, structure it, and in turn help a user in finding the right information.

Nonetheless, imagining an ideal world where the whole Web is annotated with the most exact descriptive metadata might still keep a user from finding the information he or she is looking for. This could on the one hand be due to a too vaguely defined user query, or on the other hand because of an unrealistically large number of results preventing the user from inspecting them all. One often proposed solution to this problem is personalization. Here a system tries to perform various forms of adaptation to increase the satisfaction of this particular user, based on his or her previous activities.

This problem of personalized access to external sources is becoming more and more a common one and occurs, with the current content explosion, in a widening variety of domains. Because it involves many different facets in different technical domains like data integration, user modeling, data mining, etc., as well as different involved actors (like users, metadata providers, etc.) throughout the process, providing personalized access is not an easy task. In this chapter, we try to outline a general approach to deal with this problem and practically illustrate it in the television domain. This chapter addresses research question 3 (Which generic approach has the potential to provide user-adapted data access to large heterogeneous data sources?).

4.1 A General Approach

In a variety of domains developers are struggling with the dilemma of how they can provide a more personal service to their users. For services like Amazon or Last.fm this means leading the user to the right book or music respectively. On a digital television system this can involve the suggestion
of interesting programs, while in a museum it can encompass calculating the most efficient path such that the user sees all the art he is interested in. However from a more abstract viewpoint, in all of these situations the same subproblems arise, and surprisingly similar solutions can be applied. E.g. in all of these scenarios the system needs some form of digital representation of the user to deduce exactly which books, songs, pieces of art, TV programs, etc. fit the him or her best.

As previously mentioned, the amount of data which nowadays can be found online is truly amazing, diverse and moreover, growing in every possible domain. Not only can we find a huge amount of data describing various items like books, CDs, pieces of art, etc., also more and more sources are keeping data about people and their behavior. On top of that, the combination of such sources can sometimes lead to an even bigger advantage. Having more diverse statements describing a concept, provides more raw data which a reasoner can use to deduce even more facts. It is a holistic approach since the whole after reasoning delivers more than the original parts. Moreover, since knowledge is power, a good understanding of the domain helps in providing efficient data processing strategies which ultimately can be tailored for any given user. Therefore in our approach, we strive for exploiting and combining these freely available data sources as much as possible.

In this chapter we discuss a domain-independent approach to provide user-adapted access to heterogeneous data sources. Investigating this problem, we have deduced an approach consisting of three separate components, each delivering a well-defined contribution towards this goal. In Figure 4.1 we see an overview of these three components, which we will investigate briefly here and more thoroughly in the following chapters.
4.1. A GENERAL APPROACH

4.1.1 Integration

In order to provide personalized access to a particular data set, a good knowledge of the central domain items (independent of whether we are working with books, songs, programs, art pieces, etc.) is indispensable. Hence, the more we know about the function, features and type of the items, the more raw material we have to for example compare different items, compare items with persons, deduce new information, etc. However in the real world, items often come poorly annotated. A book just having a writer and title while the topic and genre are missing, a TV program with a title and start-time but no genre, synopsis, participants or trailer, a piece of art from a famous artist but lacking the time-frame and style, etc. makes it hard to guess how related individual items are, or which for example could be interesting for a particular user. Therefore in our approach, a first important step involves increasing the knowledge of the relevant items.

In our approach, to augment the knowledge of the domain items we employ two types of publicly available information sources. The first type encompasses domain-dependent sources which can increase the quality of existing properties or add new item properties. In other words, they make the description of a given item richer. E.g. if source X describes books providing the title, author and a small synopsis while source Y provides the ISBN number, a picture of the cover and an extensive synopsis, the combination leads to a richer description of that book where metadata originated from both sources. The second type involves domain-independent sources which provide more general background knowledge modeling for example domains like language, time, geography, etc. Such sources provide knowledge which typically helps in deducing new information to compare or group different items. E.g. a source modeling ‘time’ could automatically infer that book A was published before book B, given both their dates of publishing. Or it can be deduced that the topic of A is geographically related to the topic of B, when A’s title is “Living in Beijing” while B is named “Living in Shanghai”. However, it is important to note that we do not maintain the effective content of the items like for example the text of the book, the song itself or the video material of the TV program. Rather, we integrate and manage the metadata describing those items.

Using different sources to integrate metadata is nice, however, usually different sources adopt different metadata schemas and therefore different semantics. Data integration is the field in which we try to integrate data residing in different external, and possibly heterogeneous, sources. In Figure 4.1 we see the integration step depicted in the upper left corner. To actually perform integration of various heterogeneous sources, we utilize techniques developed in the field of Semantic Web research. Via these techniques we are able to exploit publicly available knowledge structures like ontologies, vocabularies and thesauri, which usually have been developed by experts in their specific domain and can therefore be used by us to interpret otherwise meaning-less textual metadata. But also, next to schematic knowledge, we can integrate descriptive data to enrich items on instance level. In Chapter 5 we illustrate our approach towards the integration of various heterogeneous data sources, both on schema level as well as on instance level, in the television domain.

4.1.2 User Modeling

While the previous step focusses on data integration and enrichments increasing our understanding of the items, the second equally important step in the process focusses on understanding the user. Before we can adapt any kind of data processing to the user, we need to understand the user’s behavior and likings with respect to the items of the domain. And this, we can only do when we know what kind of user we are dealing with in terms of preferences, interests, demographics, background, etc.

User Modeling is the research field which tries to do exactly that. By creating a model of the user which a computer can interpret, any process can be tailored to turn it into personalized counterpart. In Figure 4.1 we see the user modeling step depicted in the upper right corner. Further, in the picture we see that a user model in our approach maintains two different types of user data, namely static and dynamic information. Static user information encompasses all
user data which remains relatively constant over longer periods of time. This data includes for example *demographics*, which comprises for example day of birth, address, education, mother tongue, cultural background, etc., *preferences*, containing for example preferred device, music loudness, screen brightness, etc., *interests*, comprised of the user’s opinion on items in the domain, and *characteristics*, including color of eyes, psychological character, time to go to bed in the evening, etc. Further, static user data can also hold different *stereotypes*, to which the user might belong. Again, static information is not 100% fixed and can evolve or change, but in general it is regarded to be rather constant as against dynamic user data.

Dynamic user data encompasses all statements which are valid within a specific *context*. Context allows us to describe in which particular situation specific user information holds. For example: a user likes to wear a green shirt, but only in the summer season, a user likes to watch the weather forecast, but mostly in the morning on working days, a user’s music taste changes depending on his location (sitting in his car, while taking a bath, during work, etc.) or watching HD content is great but not preferred on a small screen device. Therefore, dynamic user data comprises all contextualized feedback, which we modeled in our approach as an event, a user provides in response to some functionality presenting items of the domain. E.g. if the system provides the user with a set of potentially interesting items, the user can buy one (positive feedback) and remove another to the garbage bin (negative feedback). Over time, user data maintained in the dynamic part of the model can influence the static part of the user model. E.g. if the user always provides positive feedback on songs from “Mark Knopfler”, after time we can materialize this pattern in the static interests of the user. Or if the users always puts the screen brightness of his mobile to the maximum, this information can become a preference. In other words, the more a user plays and interacts with the system, the more the system can learn and deduce relevant likings, interests, preferences or patterns determining his or her behavior. How such a user model is created and maintained in our approach, will be explained in Chapter 6. Moreover, we illustrate how this concept is applied in the television domain.

4.1.3 User-adapted data access

Having a set of well-structured and enriched data items on the one hand and a user model describing the user’s context-sensitive interests, preferences, behavior, etc. on the other hand, enables the system to adapt any part of the data processing pipeline to the user. *Personalization* is the field dealing with user-centered adaptation, nowadays widely applied in many different domains. Personalization can manifest itself by for example personalizing search (by adapting e.g. a user’s search query), providing personal recommendations, filtering out inappropriate items (e.g. adult content when the user is younger than 16 years old), personalizing navigation (e.g. a personal link structure through the data), adapting the presentation of information (e.g. links to HD content are removed in case the user currently carries a small screen device), adding surprising serendipitous items (e.g. slightly unexpected items but relevant due to unusual relations), etc. In our approach, we particularly focus on providing user-adapted access to large data sources. Centrally in Figure 4.1, we see the responsible component together with some of the applied techniques towards this goal. We employ user-adapted keyword conceptualization and data semantics to enrich user queries, we filter query results by taking the user model into account and we can provide personal recommendations to find exactly those items which fit the user best in any given context.

As can be seen in Figure 4.1, the inputs of this component (the blue arrows) consist of a user request, semantically annotated data and a user model. Such a user request normally consists of the conjunction of a number of restrictions defining the item subset the user is looking for. However, since a user usually does not speak computer language, such a request can look for example like: “books written by “Raymond Feist” before 1990”. Receiving such a request, our approach first tries to filter and conceptualize keywords based on ontological information, creates a personal query in the language a computer does understand and filters the result set based on user model information. Optionally, the recommendation system calculates a score indicating how well every particular item fits the user, based on one specific or a combination of different recommendation strategies. The output of the component, which is indicated by the green arrows,
consists of an ordered list of domain items matching the user’s query. As a result, this list is, by using the aforementioned steps, adapted to the user. Therefore, if two different people enter the same textual request, different lists can be obtained. However, next to returning a list of results to the user, the adaptation process also provides a mechanism to take user feedback into account and as a result update the dynamic part of the user model. Namely, by temporarily remembering the specifics of the retrieval process and the results delivered to the user on the one hand and the feedback the user provided in response to these results on the other hand, valuable information can be gathered in the dynamic part of the user model, to further improve results in future actions.

In this thesis (and more specifically in Chapter 7) we illustrate the technique of personalization in the context of the television domain, and thereby specifically focusing on providing both user-adapted search and personal recommendations.

4.2 Conclusion

In this chapter, we investigated a potential approach to help users in finding items which suit their current expectations best, thereby answering research question 3 (Which generic approach has the potential to provide user-adapted data access to large heterogeneous data sources?). Since we anticipate that the problem of personal data access to large amounts of information will become more and more common, this proposed approach is specifically devised to serve as a solution in other domains as well. However, from here we will elucidate and illustrate every part of our approach by means of examples and/or implementations in the television domain.
Chapter 5

Integration

In the early years of the Internet, many people started to accumulate information by means of databases. Databases with data about books, CDs, people, etc., but also databases with the results from various digitalization processes of e.g. encyclopedias, digital versions of books (via scanning), etc. However, most of these data sources were private and access could only be obtained via some kind of subscription model. From a high perspective, some parallels can be drawn with the evolution of software in those days. Software, which was also almost never for free at first, evolved into free and open software when we witnessed the “for free” revolution partially initiated by the open source software movement. As this trend in software started to gain momentum, also database owners slowly started to open up their treasure coves. This to the delight of end-users who benefitted both directly and indirectly from this available data, but also many researchers and people involved in information retrieval and data mining.

Later, the next great Web evolution involved the steady transformation towards a platform based on an information sharing, interoperable and user-centered design, which is now widely known as Web 2.0. While Web 2.0 mainly focusses on community-driven user-centered applications, a huge side-effect was that people started integrating various information collections. On most Web sites geographical locations like addresses or maps have been replaced by an integrated Google Maps widget. Facebook integrated the sending and receiving of tweets (Twitter messages) and the possibility to add Flickr pictures to your profile. Numerous Web sites integrated bookmarking via del.icio.us, Yahoo, Facebook and MySpace. Flickr pictures and Wikipedia articles can be added to Google map locations. Through the provisioning of APIs, most great networks like Google, Yahoo and Facebook opened the gates towards integration, and nowadays it is impossible to surf around without seeing a single link to any of them. In general, nature always tries to accomplish tasks by spending the least amount of energy possible. Similarly, people want to accomplish as much as possible with as few clicks as possible. Integration of functionalities and data sources can help in achieving just that.

In this chapter we see how we can integrate information from several potentially heterogeneous data sources, into one data graph adhering to a well-defined schema. We illustrate these techniques in the television domain, by showing how we can enrich a television program by using different pieces of information found in different sources. This chapter provides an answer to research question 4 (How can we integrate large heterogeneous data sources into one consistent and semantically rich data model?).

5.1 Introduction

To provide context-sensitive personalized access to a specific domain, one of the most important requirements is to have richly annotated items both in terms of quality of the metadata as in the expressiveness of the metadata structure. Therefore, in Chapter 4 we saw that data integration is one of the paramount pillars of our approach. In Chapter 3, we already saw that, for the
television domain, TV-Anytime came up as the best metadata specification for TV programs, and RDF(S)/OWL came up as the best syntactical language to describe it in. However, one of the main pillars of the Semantic Web vision is that different data and knowledge sources all over the Web become even stronger when combined, because they can complement and enrich each other’s data. In this chapter, we show how integration of external sources helps in the acquisition of television program annotations. For this purpose, we focus in this chapter on two different aspects of integration:

- Schema integration: Over the years, knowledge structures like ontologies, vocabularies, thesauri, etc. were developed by people or groups who are experts in specific domains. Usually, these structures are completely independent like e.g. an ontology modeling time or a thesaurus modeling the English language. However, in most cases relying on such knowledge can help enormously in modeling general concepts consistently. Therefore, by integrating their concepts and semantics into the domain model, their conceptualizations can be reused.

- Instance data integration: Maintainers of metadata sources usually accumulate data about instances with a certain goal in mind. They model their data from a certain perspective ultimately serving their own goals. Amazon for example keeps different metadata about a book than the ISBN database does. The nice thing however is that when we integrate metadata from different sources the total description of an instance becomes richer than the description of any of the sources individually.

However, as explained in Chapter 3, the integration of different sources depends heavily on the ability to uniquely identify objects to conclude for example that object $O$ in source $S$ is exactly the same as object $O'$ in source $S'$. We also saw that to effectuate this match, a representation in RDF easily gets the upper hand over an XML-based representation. In the following section we discuss the translation of the TV-Anytime XML specification into RDF(S)/OWL.

## 5.2 TV-Anytime in RDF(S)/OWL

### 5.2.1 Metadata Schemas

To transform the TV-Anytime specification to RDF(S)/OWL we make use of the ReDeFer\(^2\) approach which describes among others the XSD2OWL and XML2RDF tools [100]. XSD2OWL is able to transform an XML Schema Document (XSD) into an OWL file, while the XML2RDF tool translates the associated XML files to RDF given the OWL ontologies generated by XSD2OWL. The XSD2OWL tool transformations are executed via an Extensible Stylesheet Language Transformation (XSLT). This approach has previously been demonstrated with the translation of the MPEG-7 specification into OWL as described in [101]. However, we will see later that this approach also engenders some disadvantages.

In the following example we see a part of the TV-Anytime XSD specification, showing the original TV-Anytime CreationCoordinatesType structure which models where and when a television program was created:

```xml
<complexType name="CreationCoordinatesType">
  <sequence>
    <element name="CreationDate" type="tva:TVATimeType" minOccurs="0"/>
    <element name="CreationLocation" type="mpeg7:regionCode" minOccurs="0"/>
  </sequence>
</complexType>
```

In this description we see that the program’s creation coordinates can be described by a date and a location, where both are optional. The creation date uses the TV-Anytime TVATimeType

\(^1\)http://isbndb.com/

\(^2\)http://rhizomik.net/redefer/
type to model the time of creation by a mandatory \texttt{TimePoint} and an optional \texttt{Duration} (the default value for \texttt{maxOccurs} and \texttt{minOccurs}, expressing the cardinality, is 1). The three MPEG-7 references (\texttt{regionCode}, \texttt{timePointType}, \texttt{durationType}) point to MPEG-7 simple types. The TV-Anytime \texttt{TVATimeType} and MPEG-7 \texttt{regionCode} for example look like:

\begin{verbatim}
<complexType name="TVATimeType">
  <sequence>
    <element name="TimePoint" type="mpeg7:timePointType"/>
    <element name="Duration" type="mpeg7:durationType" minOccurs="0"/>
  </sequence>
</complexType>

<simpleType name="regionCode">
  <restriction base="string">
    <whiteSpace value="collapse"/>
    <pattern value="[a-zA-Z]{2}(-[a-zA-Z0-9]{1,3})?"/>
  </restriction>
</simpleType>
\end{verbatim}

When we translate the TV-Anytime \texttt{CreationCoordinatesType} to an OWL file via the XSD2OWL tool, the following code is automatically generated (do note that some of the syntax is shortened due to space constraints:

\begin{verbatim}
<owl:Class rdf:ID="CreationCoordinatesType">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="#CreationDate"/>
          <owl:allValuesFrom rdf:resource="#TVATimeType"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#CreationDate"/>
          <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#CreationLocation"/>
          <owl:allValuesFrom rdf:resource="&mpeg7;regionCode"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#CreationLocation"/>
          <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
\end{verbatim}
However, there is a significant problem with this approach, which is actually due to OWL's expressiveness, or at least the lack thereof. The OWL specification is unable to define or restrict custom data types [173], and therefore still relies on external data types from for example XML or MPEG-7. If we look at MPEG-7's `MPEG7:regionCode` and TV-Anytime's `TVA:CRIDType`, we see that they are defined as:

```xml
<simpleType name="regionCode">
  <restriction base="string">
    <whiteSpace value="collapse"/>
    <pattern value="[a-zA-Z]{2}(-[a-zA-Z0-9]{1,3})?"/>
  </restriction>
</simpleType>
```

```xml
<simpleType name="CRIDType">
  <restriction base="anyURI">
    <pattern value="(c|C)(r|R)(i|I)(d|D):/.*"/>
  </restriction>
</simpleType>
```

These two XML examples are defined by means of regular expressions which makes it impossible to mimic their behavior in OWL classes as the OWL specification does not allow for this. As a result, the XSD2OWL tool cannot translate these simple types and simply replaces them by the XML string type. For example, the TV-Anytime's CRID reference type `TVA:CRIDRefType`:
5.2. TV-ANYTIME IN RDF(S)/OWL

```xml
<complexType name="CRIDRefType">
  <attribute name="crid" type="tva:CRIDType" use="required"/>
</complexType>
```

is translated into:

```xml
<owl:Class rdf:ID="CRIDRefType">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#crid"/>
      <owl:allValuesFrom rdf:resource="&xsd;String"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#crid"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

As shown, the property `crid` in OWL can only take instances of the class ‘String’, leading to a loss of semantics. To solve this issue, we generated a TV-Anytime data type file containing all the TV-Anytime `simpleTypes` which cannot be expressed in OWL. This allows us to refer to these types in the same manner as we refer to for example XML data types. The restriction above now becomes:

```xml
<owl:Class rdf:ID="CRIDRefType">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#crid"/>
      <owl:allValuesFrom rdf:resource="&tvaTypes;CRIDType"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  ... 
</owl:Class>
```

Notable is that the OWL syntax is much more verbose in expressing the same statements than the original XML. This is largely due to the fact that with all advantages that OWL brings, also comes the effort in staying backwards compatible with the original RDF(S) syntax [127]. Combining the already existing MPEG-7 OWL version with this TV-Anytime OWL description now provides us with a fully functional OWL version of TV-Anytime. This structure can now, following the basics of the OWL language, support reasoning over classes (e.g. by subclass relations) and properties (e.g. via properties introducing transitivity, inverses, etc.), but also facilitate easy integration of external sources as we will see in following sections.

5.2.2 Classification Schemes

Besides the general TV-Anytime metadata specification (described in the previous subsection), TV-Anytime also defines a set of Classification Schemes (CS). These schemes are modeled as term hierarchies, a concept which was already explained in Chapter 3.4. In [101], Garcia et al. also briefly mention the CS2OWL tool specifically made to translate these kinds of classifications into an OWL representation. Taking again the example defined in Figure 3.7, we see how a small extract of the content CS is defined in TV-Anytime:
CHAPTER 5. INTEGRATION

When applying the CS2OWL tool, this small code snippet translates into:

```xml
<owl:Class rdf:about="&genre;3.1">
  <rdfs:label xml:lang="en">NON-FICTION</rdfs:label>
</owl:Class>
<owl:Class rdf:about="&genre;3.1.1">
  <rdfs:label xml:lang="en">News</rdfs:label>
  <rdfs:comment xml:lang="en">Time-sensitive information</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&genre;3.1"/>
</owl:Class>
<owl:Class rdf:about="&genre;3.1.1.1">
  <rdfs:label xml:lang="en">Daily News</rdfs:label>
  <rdfs:comment xml:lang="en">Regular program carrying breaking and current...</rdfs:comment>
  <rdfs:subClassOf rdf:resource="&genre;3.1.1"/>
</owl:Class>

...<rdf:RDF>
  <owl:Class rdf:about="&tva;GenreType">
    <rdfs:label xml:lang="en">Genre class</rdfs:label>
  </owl:Class>
  <rdf:Description rdf:about="&cs;3.1">
    <rdf:type rdf:resource="&tva;GenreType"/>
  </rdf:Description>
  <rdf:Description rdf:about="&cs;3.1.1">
    <rdf:type rdf:resource="&tva;GenreType"/>
  </rdf:Description>
  <rdf:Description rdf:about="&cs;3.1.1.1">
    <rdf:type rdf:resource="&tva;GenreType"/>
  </rdf:Description>

In this example we see that the CS2OWL tool transforms every genre term into a class and describes the hierarchy by means of the rdfs:subClassOf relation. This modeling choice is of course valid, however, modeling such a term as a class feels strange. Moreover, what could be in this case an instance of the class genre:3.1.1? It feels more natural to have one class modeling a ‘genre’ in general, where every specific genre defined in the genre hierarchy becomes an instance of this genre class. Consequently, in this model the genre ‘news’ (which is represented by the term genre:3.1.1) becomes an instance of the ‘genre’ class. Adopting this model, translates our example into:

```xml
<owl:Class rdf:about="&tva;GenreType">
  <rdfs:label xml:lang="en">Genre class</rdfs:label>
</owl:Class>
<rdf:Description rdf:about="&cs;3.1">
  <rdf:type rdf:resource="&tva;GenreType"/>
</rdf:Description>
<rdf:Description rdf:about="&cs;3.1.1">
  <rdf:type rdf:resource="&tva;GenreType"/>
</rdf:Description>
<rdf:Description rdf:about="&cs;3.1.1.1">
  <rdf:type rdf:resource="&tva;GenreType"/>
</rdf:Description>
</rdf:RDF>

...
While this way of modeling feels more natural in structure, it also bears a consequence. For one, the explicit hierarchy is lost since we cannot use \texttt{rdfs:subClassOf} between instances anymore (note that this would become possible if we would use OWL Full instead of OWL-DL, since OWL full allows an individual to be regarded as a class). However, since we strive for modeling our data model in at most OWL-DL (to retain computational completeness and decidability), we cannot use OWL Full. Another way of solving this issue comes by making use of \textit{SKOS} relations as explained in the following section.

## 5.3 Schema Integration

The generated OWL representation of TV-Anytime solves some of our previously stated shortcomings of the XML version. Every instance of this schema now behaves as a labeled directed graph where every node (resource) is uniquely identified, and possibly connected to other nodes via (also uniquely defined) properties. Let us take a look again at program \textit{P}, which was introduced in Chapter 3, and can be seen here in Figure 5.1. However, this time \textit{P} is described by an RDF instance of the OWL version of the TV-Anytime specification (\textit{P}'s description among others also contains an instance of the TV-Anytime class \texttt{TVA:CreationCoordinatesType} which was used above to illustrate this translation).

However, one of the shortcomings described in Chapter 3, among which for example the fact that some strings like “2004-04-15T00:00:00” or “BE-VAN” (the official \texttt{regionCode} of “Antwerp, Belgium” as described by ISO 3166-1 and ISO 3166-2) bear no semantics when modeled as strings (albeit well-defined strings), is still not solved here. In the following subsection we will show that integration of different external sources can help in providing additional meaning to otherwise flat strings or badly defined concepts, and thereby increase the semantic power of this TV-Anytime description. By doing so, we can tackle the issues previously described.

### 5.3.1 Addition of temporal and spatial semantics

Previously, we have shown, with examples of the MPEG-7 data types \texttt{regionCode} (having values like “BE-VAN”) and \texttt{timePointType} (having values like “2004-04-15T00:00:00”), that some values of properties are modeled as strings and therefore limit the exploitation of their meaning. Just like in the case of XML, if we want to exploit temporal values like “2004-04-15T00:00:00” smartly (to extract for example that one program start before another one), we need somewhere an interpretation step parsing this string. A query asking for programs broadcast on Saturday starting between 8pm and 9pm would require that the query language is able to interpret these time strings in order to return the answer. While this is not unthinkable in case of time-related strings (the SQL query language understands some time formats), it becomes more problematic when we look for example at region codes. We can not expect that a query language understands “BE-VAN” and ‘knows’ that this is a location in Belgium.

In the context-sensitive television domain, time and location are extremely important. A user for example might not be interested so much in a long late-night movie on working days, while he is on Friday and Saturday. Or a user can be a huge fan of movies produced in Belgium and neighboring countries, but not so much of American films. If we could create concepts of values like “2004-04-15T00:00:00” or “BE-VAN” we could relate them in turn to other concepts allowing new relations to be found or deduced like e.g. that “2004-04-15T00:00:00” comes before “2005-04-16T00:00:00” or that ‘VAN’ denotes a part within ‘BE’.

**Obtaining a rich temporal description with OWL-Time**

OWL-Time (or the Time ontology in OWL), is an ontology developed to describe the temporal content of Web pages and the temporal properties of Web services [126]. This ontology covers topological properties of \textit{instants} and \textit{intervals}, measures of \textit{duration}, and the meanings of clock and calendar terms. Therefore, it is exceptionally useful as a means to add semantics to otherwise
Figure 5.1: Example of a TV-Anytime described program in RDF

more static expressions of time. The main classes in OWL-Time include Instant (a specific point in time), ProperInterval (a time interval), DurationDescription (description of the duration of an interval) and DateTimeDescription (description of a specific point in time). In the following example we see how the TVA:TimePoint property of the TVA:TVATimeType class would look when described in OWL-Time (‘TIME’ is the abbreviation of the OWL-Time namespace):
5.3. SCHEMA INTEGRATION

In this description we see that not only every part of the date string is parsed into different meaningful properties, but also that new knowledge was gained through properties like `TIME:dayOfWeek` and `TIME:dayOfYear`. Obviously, such a description is much more useful in terms of querying and reasoning than the original time string. Therefore, if the `TVA:TVATimeType` class would have an `OWL:ObjectProperty TVA:TimePoint` with the range `TIME:DatetimeDescription` instead of the current `OWL:DatatypeProperty` with the range `mpeg7:timePointType`, we can enable this technique in TV-Anytime. However, we cannot just change the schema of the TV-Anytime specification to our liking, due to reasons of backward compatibility. In such a case, existing TV-Anytime instantiations would suddenly not adhere to the schema anymore. As solution we therefore introduce a new class which is a subclass of the original `TVA:TVATimeType` class. This class can then be adapted accordingly without breaking the compatibility with older TV-Anytime instances. This new class bears the same name, but is defined in our own iFanzy namespace:

```xml
<owl:Class rdf:ID="TimeType">
  <rdfs:subClassOf rdf:resource="&tva;TVATimeType"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#TimePoint"/>
      <owl:allValuesFrom rdf:resource="&time;DatetimeDescription"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

The new `IFZ:TimeType` class inherits all semantics from the original TV-Anytime class, but also adds an extra property `IFZ:TimePoint` which has the `TIME:DatetimeDescription` class as its range. By doing so, this class is still a valid TV-Anytime class as it extends the original one. It just adds a property allowing us to describe the time point not just as a `mpeg7:timePointType` but as a `TIME:DatetimeDescription` from the OWL-Time ontology as well. Of course this class still needs to be instantiated with the correct values. This instantiation is done when an original XML TV-Anytime instance is converted to RDF. At that time, we calculate the correct instance of the `IFZ:TimeType` class based on the original time string. Having this new instance, we can now for example perfectly execute queries asking for all programs broadcast on Saturday between 8pm and 9pm without an extra interpretation layer.

However, we do need to take a subtle hidden tradeoff into account. Modeling time related properties by means of OWL-Time classes indeed provides more query flexibility, however it also contributes to a considerable increase of triples, making the database larger and more cumbersome. On the other hand not including this integration, makes that the query language must support and interpret date time formats like “2004-04-15T00:00:00”, which can influence query performance.
Adding spatial semantics

In TV-Anytime a region code is used to uniquely identify locations used to express a program’s production location, creation coordinates, etc. This region code follows the ISO 3166-1 and ISO 3166-2 specifications. For example “BE-VAN” (the official region code of “Antwerp, Belgium”) consists of two parts, namely ‘BE’ which is the code for ‘Belgium’ according to ISO 3166-1 and ‘VAN’ which translates to ‘Antwerp’ according to ISO 3166-2. While there thus exists some further meaning in this at first sight simple description, it requires knowledge of these ISO standards. It is now for example impossible to describe ‘smaller’ locations like villages, streets and parks or bigger locations like continents. Nor is it possible to use this in order to find for example locations to the north of Antwerp, or countries to the south of Belgium. In other words, there is no way to reason over location hierarchies or relative positioning without an extra interpretation layer.

Online, there are several sources of geographical information freely available. Among them the National Geospatial-intelligence Agency (NGA) which provides the GEOnet Names Server\(^3\) (GSN). GSN provides database access to foreign geographic feature names and serves as the official repository of foreign place-name data, approved by the US Board on Geographic Names (BGN). Approximately 20,000 of the database’s features are updated monthly while it contains 4.0 million features with 5.5 million names approximately.

Furthermore, besides central sources, there also exist sources listing geographical information for one specific country. The U.S. Geological Survey Geographic Names Information System or GNIS for example, provides geographical information solely about the USA. GeoBase\(^4\) on the other hand, is a federal, provincial and territorial government initiative maintaining a base of quality geospatial data for all of Canada.

3\(http://earth-info.nga.mil/gns/html/index.html\)
4\(http://www.geobase.ca/\)

Figure 5.2: GeoNames location instance in RDF
However, over the years, there was an initiative to integrate data from various geographical sources (like GNS, GNIS, GeoBase, etc.) into one large consistent database called GeoNames\(^5\). The GeoNames geographical database is available free of charge under a creative commons attribution license. It contains over eight million geographical names and consists of 6.5 million unique features of which 2.2 million populated places and 1.8 million alternate names. All features are categorized into one out of nine feature classes and further subcategorized into one out of 645 feature codes. GeoNames integrates geographical data such as names of places in various languages, elevation, population and others, from various online sources. All latitude and longitude coordinates are in WGS84 (World Geodetic System 1984).

Furthermore, a very nice feature of GeoNames is its open mindedness towards new technologies. GeoNames has embraced the Semantic Web to open up its vast database to the public. Moreover, an OWL ontology was defined to fix the structure of their database in both OWL lite and OWL full. Every single geographical location is uniquely identified with a URI, and wherever possible, links are provided to Wikipedia. In Figure 5.2 we see the GeoNames RDF instance of “Antwerp, Belgium” (do note that we use the ‘geo’ abbreviation to reduce the size of the presented code). In this example, we see alternative names in other languages, a feature class (\(P\) denotes “City, village,...”), a feature code (\(P.PPL\) denotes “a city, town, village, or other agglomeration of buildings where people live and work”), the country, population, longitude, latitude, parent feature and nearby features (as reference to another RDF description), a Google map reference, Wikipedia articles, and mappings to other RDF data sources via the \texttt{owl:sameAs} construct.

Next to GeoNames, there also exists another large geographical thesaurus, called the Getty Thesaurus of Geographic Names\(^6\) or TGN. TGN is a structured vocabulary providing geographical information. It contains around 1.1 million names, place types, coordinates, and descriptive notes, focusing on places important for the study of art and architecture. Despite its focus of one specific domain, TGN still contains a huge amount of useful geographical information. Via narrower, broader and parent-child relations, TGN maintains an extensive hierarchy of places around the world. On the other hand, the particularly strong focus on places relevant to art and architecture, implies that for places less interesting for art and architecture, less details are available. A second disadvantage of this vocabulary is the fact that in order to use the data a license must be signed and required fees must be payed. While they do provide a free Web service on their Web page, it makes it hard to integrate data into our data graph. However, the MultimediaN E-Culture project\(^7\), which is related to ours but focusses on the arts and culture domain, opened a public query endpoint providing access to their RDF knowledge database. This knowledge structure contains among others the Getty thesauri which were converted from their original XML format into an RDF/OWL representation [18, 210]. Although the Getty thesauri are licensed, they support limited research and cataloging efforts in non-commercial environments.

In Figure 5.3 we see a shortened RDF description of the concept describing the location “Antwerp, Belgium” in TGN. While there is some clear overlap between TGN and GeoNames (e.g. label, latitude, longitude, etc.), there are also some differences. GeoNames contains a link to a map, features classes and Wikipedia links, while TGN on the other hand contains a vast list of alternative names, a longer textual description and a more grounded set of semantical relations like broader, transitive broader, semantically related, etc. Together, GeoNames and TGN complement each other perfectly, due to their different points of view. Moreover, both are available in RDF, making it even easier to integrate their knowledge. Unfortunately, both of these descriptions do not contain the region code property (as used by TV-Anytime), which makes it impossible to immediately see that the resource modeling ‘Antwerp’ is the same individual as defined by the string “BE-VAN”. Still, the GeoNames Web site contains this information\(^8\), they just did not include it in the RDF. However, this information is essential to make the connection between the region code and more structured descriptions. Therefore, we introduced our own \texttt{IFZ:Location} class in the iFanzy namespace, which defines a property \texttt{IFZ:regionCode}.

---

\(^5\)http://www.geonames.org/

\(^6\)http://www.getty.edu/research/tools/vocabulary/tgn/index.html

\(^7\)http://e-culture.multimedian.nl/

\(^8\)In “http://www.geonames.org/BE/administrative-division-belgium.html” we see the Belgian codes
in which links to the GeoNames and TGN resources can then be facilitated by the OWL:sameAs property. Linking region code strings to the correct GeoNames and/or TGN concepts is performed via the following procedure: 1) From the country part (here ‘BE’), we can find the country link on GeoNames (here “http://www.geonames.org/countries/BE/belgium.html”); 2) Every GeoNames country Web page contains a link to its administrative divisions page (here “http://www.geonames.org/BE/administrative-division-belgium.html”); 3) On that page we find a list of all divisions together with a link (here ‘VAN’ is listed as reference to ‘Antwerp’); 4) This division link then forwards us further to the RDF page of that division (here ‘Antwerp’ points to “http://sws.geonames.org/2803138/”). Although this strategy of following links might seem cumbersome, it does deliver a URI to an RDF description of any region code. The iFanzy instance of feature “Antwerp, Belgium” now becomes:

```xml
<Location rdf:about="http://ifanzy.nl/Ontologies/Geo/Instances#loc_1">
  <regionCode>BE-VAN</regionCode>
  <owl:sameAs rdf:resource="http://sws.geonames.org/2803138/">
    <owl:sameAs rdf:resource="http://e-culture.multimedian.nl/ns/getty/tgn#7007856"/>
  </owl:sameAs>
</Location>
```

This new iFanzy geographical resource is an instance describing the location “Antwerp, Belgium”, including a link to the GeoNames feature ‘2803138’ and the TGN resource ‘7007856’, however with an additional property IFZ:regionCode. Now, to be able to deploy this newly defined location class in for example the TVA:CreationCoordinatesType class of the original television program $P$, we use the same technique as explained in the previous section. By subclassing the original TV-Anytime TVA:CreationCoordinatesType class in our own iFanzy namespace, we can add an extra property as shown in the following example:
5.3. SCHEMA INTEGRATION

All the original instances of the TV-Anytime TVA:C\text{reation}CoordinatesType class are now also instances of the iFanzy IFZ:C\text{reation}CoordinatesType. However, the iFanzy incarnation of the creation coordinates class also allows to make a reference to an iFanzy location containing the region code property as well as a potential connection to GeoNames, TGN and/or any other RDF/OWL resource. When TV-Anytime instances are retrieved (containing an official region code like “BE-VAN”), we can automatically add a property to the iFanzy location linked to the correct GeoNames/TGN instances via the OWL:sameAs property. Thanks to this semantic enrichment we can now infer that this program was created in Belgium, something which was previously impossible without an extra interpretation of the “BE-VAN” string.

Another approach towards the integration of geographical data and generating ontologies, is the Geographic Knowledge Base (GKB) [61]. However, in order for GKB to be able to perform these integrations, several requirements need to be met. In comparison, we can integrate no matter which class thanks to the OWL:sameAs connection property.

5.3.2 Addition of Lexical semantics

Most of the time, when people use search engines to find information, a sequence of keywords is phrased. Finding documents matching these keywords usually involves indexing all textual information of these documents and returning those which have a positive match on those keywords. However, it can occur that a person is searching for a term, e.g. car, and no results are found while there exist documents that mention the word auto or automobile (adding to the false negative rate). In other words, while there might exist potentially good results in the document set, none were returned because no exact match was found. As a consequence, this would severely limit the user’s ability to search for content items in an accurate way.

WordNet is an online lexical database designed for use under program control [170]. WordNet is organized around the concept of synsets or synonym sets. English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms, each representing a lexicalized concept. Between these synsets various semantic relations are defined, linking them together [171]. The most relevant links are defined in [170] as:

- Synonymy is WordNet’s basic relation, because WordNet uses sets of synonyms (synsets) to represent word senses. A synonym relation for example exists between ‘car’ and ‘auto’.
- Antonymy (opposing-name) is also a symmetric semantic relation between word forms, especially important in organizing the meanings of adjectives and adverbs. An antonym relation exists for example between ‘carnivorous’ and ‘herbivorous’, which are each others opposites.
- Hyponymy (sub-name) and its inverse, hypernymy (super-name), are transitive relations between synsets. Because there is usually only one hypernym, this semantic relation organizes the meanings of nouns into a hierarchical structure. E.g. ‘ambulance’ is a hyponym of ‘car’ and ‘car’ a hypernym of ‘ambulance’.
• Meronymy (part-name) and its inverse, holonymy (whole-name) describe the semantic relations that hold between a part and the whole. WordNet distinguishes component parts (e.g. ‘suspension’ is a part meronym of ‘car’), substance parts (e.g. ‘snowflake’ is a substance meronym of ‘snow’) and member parts (e.g. ‘locomotive’ is a member meronym of ‘train’).

• Troponymy (manner-name) describes the semantic relation of being a manner. It is for verbs what hyponymy is for nouns. E.g. ‘whispering’ is a troponym of ‘talking’ (whispering is a manner of talking), ‘running’ is a troponym of ‘walking’ (running is a manner of walking).

• Entailment relations show that something is inferred, deduced or implied. For example, the verb ‘to snore’ entails the verb ‘to sleep’.

In 2006, WordNet was introduced in the Semantic Web world by an RDF/OWL version. Two different versions were created, namely WordNet Basic and WordNet Full. The reason for that was to reduce the magnitude (influencing the memory footprint) of the ontology if not all functionality was required. WordNet Basic therefore does not contain any Words and WordSenses and only contains the synsets. The full WordNet schema has three main classes: Synset, WordSense and Word, where each instance of these classes has its own URI which for example looks like:

http://www.w3.org/2006/03/wn/wn20/instances/synset-car-noun-1
http://www.w3.org/2006/03/wn/wn20/instances/wordsense-car-noun-1
http://www.w3.org/2006/03/wn/wn20/instances/word-car

The first URI identifies the ‘synset’ instance for the word ‘car’. This synset contains a number of synonymic word senses, among which we see ‘car’, ‘auto’, ‘automobile’, etc. The second URI identifies the first WordSense contained in that synset. The WordSense instance of the word ‘car’ in turn points to the instance modeling the general word ‘car’ which can be seen in the third URI.

In our application, we exploit lexical semantics to improve the keyword-based search in television programs. The program described in Figure 5.1 contains a title, synopsis and associated keywords. All of these fields are eligible for being used in a keyword-driven search. There are basically two WordNet relations which are valuable for keyword search: synonymity (‘auto’ or ‘automobile’ can also be relevant when searching for ‘car’) and hyponymity (‘bus’, ‘convertible’, ‘limousine’, etc. are types of cars in WordNet and thus relevant for the search). To make a mapping from the keywords in the TV-Anytime ontology, to words in WordNet we again use the OWL:sameAs property. The TVA:KeywordType in Figure 5.1 for example becomes:

```xml
<KeywordType rdf:ID="KW_964_1">
  <Type>main</Type>
  <Value rdf:datatype="&xsd;String">murder</Value>
  <owl:sameAs rdf:resource="http://www.w3.org/2006/03/wn/wn20/instances/word-murder"/>
</KeywordType>
```

5.3.3 SKOS enrichment of Classification Schemes

The Simple Knowledge Organization System or SKOS, provides a model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, folksonomies, and other similar types of controlled vocabulary [137]. SKOS is entirely modeled in RDF and consists of a central class SKOS:Concept and a whole multitude of both labeling and semantic relation properties. Looking at SKOS’s classes and properties, it might look like another formal knowledge representation language like for example OWL. However, SKOS was modeled to help express structures like thesauri, classification schemes, etc. which inherently do not contain formal semantics. Therefore, SKOS should be seen as a useful toolkit developed to represent these thesauri and classification schemes as independent data models.

http://www.w3.org/TR/wordnet-rdf/
5.3. SCHEMA INTEGRATION

Previously in subsection 5.2.2, we saw the translation of the TV-Anytime Classification Schemes. However, the (preferred) instance-based translation of the hierarchy showed some disadvantages. Some relevant properties to express e.g. the hierarchy, the name and definition field, etc. of a classification term were missing. SKOS however, provides exactly those constructs needed to tackle these issues. To introduce the hierarchy in the descriptions, SKOS provides **SKOS:narrower** and **SKOS:broader** to say that one concept is more refined or more general respectively. To model the name and definition properties of the hierarchy, we can make use of SKOS’s collection of annotation properties like: **SKOS:prefLabel** (preferred label), **SKOS:altLabel** (alternative label), **SKOS:definition**, etc. The updated translation example used in subsection 5.2.2 can be seen in Figure 5.4 (do note, some URIs are shortened due to space constraints). The newly introduced genre class **IFZ:GenreType** in the iFanzy namespace inherits from the original TV-Anytime class **TVA:GenreType** and the SKOS class **SKOS:Concept**.

```xml
<rdf:RDF>
  <owl:Class rdf:about="GenreType">
    <rdfs:subClassOf rdf:resource="&tva;GenreType"/>
    <rdfs:subClassOf rdf:resource="&skos;Concept"/>
    <rdfs:label xml:lang="en">Genre class</rdfs:label>
  </owl:Class>

  <rdf:Description rdf:about="3.1">
    <rdfs:label xml:lang="en">NON-FICTION</rdfs:label>
    <skos:narrowerTransitive rdf:resource="#3.1.1"/>
  </rdf:Description>

  <rdf:Description rdf:about="3.1.1">
    <rdfs:label xml:lang="en">Time-sensitive information</rdfs:label>
    <skos:broaderTransitive rdf:resource="#3.1.1.1"/>
  </rdf:Description>

  <rdf:Description rdf:about="3.1.1.1">
    <rdfs:label xml:lang="en">Regular program carrying...</rdfs:label>
    <skos:broaderTransitive rdf:resource="#3.1.1"/>
    <skos:narrowerTransitive rdf:resource="#3.1.1.1"/>
  </rdf:Description>
</rdf:RDF>
```

Figure 5.4: Part of the updated genre hierarchy

The newly introduced properties **SKOS:narrower** and **SKOS:broader** indeed restore the hierarchy between the different terms. This in turn enables us again to reason over the terms and their meaning to one another. Searching for ‘News’ programs should also include searching for “Sport news”, “Daily news”, “Business news”, “Traffic news”, etc. by reasoning that these genres are narrower than the initial ‘News’ genre. The broader/narrower hierarchy allows just that. Do note that we prefer using **SKOS:broaderTransitive** and **SKOS:narrowerTransitive**, over **SKOS:broader** and **SKOS:narrower**. The reason, obviously, is that **SKOS:broader** and **SKOS:narrower** are not specifically defined to be transitive by themselves. However, we definitely want of be able to reason that genre “Daily News” (3.1.1.1) is indeed also narrower than the genre “NON-FICTION” (3.1).

In Figure 5.5, we see a representation of the TV-Anytime genre hierarchy. At the top we have the general root which has a number of children, each indicating a subgroup of genres like sports, fiction, non-fiction, etc. As previously seen, the tree itself is built by mean of the SKOS broader/narrower relations. These relations relate genres on every path from the root to a leaf.
However, sometimes it can occur that two genres which are not connected via broader/narrower relations (even not through a chain of these properties) still seem to be related. In Figure 5.5 we show two such cases. In the first case (indicated by the blue arc), two genres from different genre groups (in this case non-fiction and sports) are related. Their only possible relation in the tree is that they have a common ancestor (the root itself). However, since this is true for every two arbitrary genres in the hierarchy, this is a very weak relation. In the second case (indicated by the red arc), two genres from the same genre group (in this case sports) seem to be related. To materialize these kinds of relations, we can use the SKOS:related relation which can exist between two SKOS:Concept classes.

Both these hierarchical and relatedness properties are important because they can directly define differences or similarities between television programs. Moreover, more relations between terms directly contribute to the insights of how programs relate to each other. Therefore, they prove invaluable in for example clustering, personalization or data mining algorithms. Examples of related terms residing in different genre groups (examples of the blue arc) include for example (the concept’s label is shown in square brackets):

- ‘3.1.1.9’ [sports](news with topic) and ‘3.2’ [sports]
- ‘3.1.9.4’ [musical](news coverage of) and ‘3.4.10’ [musical]
- ‘3.8.5.1’ [fishing](the hobby) and ‘3.2.6.5’ [fishing](the sport)

Similarly, there are also terms within the same genre group which are related (examples of the red arc):

- ‘3.1.1.10.1’ [Arts](news coverage) and ‘3.1.4’ [Arts]
- ‘3.1.1.10.3’ [Film](news) and ‘3.1.4.5’ [cinema](world of film)
- ‘3.2.6.2’ [Yachting] and ‘3.2.8.2’ [Motor boating]

Determining these relations is a semi-automatic process. The automatic part involves a check for partial string overlap, (overlap between the names of two terms) which for example would return ‘sports’ (news of) and ‘sports’ as an eligible pair to be connected by the SKOS related property. Further, also WordNet relations can help in this process by providing for example that ‘film’ and ‘cinema’ are synonyms as they reside in the same synset. However, these automatic relation detection measures can also suggest pairs which are absolutely not related at all. For example the genre ‘witchcraft’ which is a type of religion in the genre tree non-fiction could falsely be matched to ‘craft’ which is a genre in the hobby/leisure group. Therefore, an expert’s involvement can be indispensable to both maybe discover other relevant relations or to at least detect errors from the automatic process.
5.3. SCHEMA INTEGRATION

5.3.4 The iFanzy Domain Model Overview

In Figure 5.6, we see an illustrative part of the iFanzy conceptual model depicting the different constructs described earlier in this section. So far we discussed the integration with external sources like OWL-Time, TGN, WordNet, GeoNames, etc., together with their connection to classes in the RDF(S)/OWL version of TV-Anytime, currently used as central domain model in iFanzy.

In this figure we see three strongly connected layers with the iFanzy layer (containing iFanzy specific classes) residing in the middle. The top layer shows existing ontology classes from which iFanzy classes inherit semantics. E.g. the iFanzy IFZ:Program (where ‘IFZ’ is the abbreviation of the iFanzy namespace) concept inherits from the TV-Anytime TVA:ProgramInformationType class obtaining properties like a program id, a basic description class (containing most of the data type properties), etc. As can be seen, the iFanzy IFZ:GenreType inherits from both the respective TV-Anytime class as well as from the SKOS concept class. Next to inherited properties, the iFanzy program class has a set of additional properties like IFZ:remakeOf and IFZ:followedBy, which are both mainly used for relations between movies. Also, relations defined by the Dublin Core (DC) specification\(^\text{10}\) like DC:relation can be used here to denote related programs. Clearly, subclassing existing ontological classes introduce additional semantics, which in turn allow for richer instances of the iFanzy program class. In the bottom layer, we see concepts of external ontologies like OWL-Time, TV-Anytime, geographical classes and WordNet which are refer-

---

\(^{10}\)http://dublincore.org/
enced from classes in the iFanzy layer. For example the IFZ:TimeType class (which inherits from the TVA:TVATimeType class) has an extra property pointing to the TIME:DatetimeDescription class as previously explained. The IFZ:DurationType follows a similar approach pointing to the TIME:DurationDescription. In Figure 5.6 we also see the WordNet class WN:Word in the bottom right corner. However, this WordNet class is not connected to any other class in the schema. This is because the relation between a TVA:KeywordType and a WN:Word is not straightforward. They are not equivalent classes and neither one is a subclass of the other (some keywords are not words e.g. “Pure action”). Instead of being connected on the schema level, they are connected on the instance level. Every keyword which is found to be a valid word in the WordNet thesaurus, is connected to that word via a OWL:sameAs relation. Further we also see the geographical classes TGN:Place and GEO:Feature, which are equivalent classes of the iFanzy IFZ:Location class.

With this TV-Anytime based domain model, enriched with semantics from various other external knowledge repositories like OWL-Time, WordNet, TGN, GeoNames, SKOS, Dublin Core, etc., we believe we obtained a very expressive and flexible model which lends itself perfectly to annotate a myriad of different kinds and types of television programs. Further, this model now fully complies with the requirements set to support context-sensitive personalized access to television content, as outlined in Chapter 3. In the following section, we discuss how actual program metadata can be obtained and fitted within this model.

5.4 Instance data integration

The Semantic Web’s main goal is to build a global Web of machine-readable data. However, besides the fact that this goal is very ambitious, it is also unrealistic to realize from one day to the next. While most are convinced that it would be a great improvement to the Web’s structure, it also demands a lot of devotion and commitment to achieve a small step toward this greater goal. The Linked Open Data Community Project\(^{11}\) works to do exactly that. While the Semantic Web, or Web of Data, is the goal or the end result, Linked Data provides the means to reach that goal [39]. The idea of Linked Data is that when different people make ontologies, grasping the knowledge they personally excel in, others can reuse this knowledge or refer to it by exploiting the uniqueness of a URI. The Semantic Web isn’t just about putting data on the Web. It is about making links, so that a person or machine can explore the Web of data [35]. This methodology of trying to connect as much different knowledge repositories as possible, is also known as the linked data initiative [35]. In [35], Berners-Lee further outlines four basic rules for publishing data on the Web, assuring that all published data integrates into one global data space:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
- Include links to other URIs, so that they can discover more things.

While it is indeed important to use proper URIs (bullet one and two) and to provide links to different sources (bullet four), the whole network stands or falls with accessibility of the sources and the quality of the data. The more sources that open up their vast data collections and do so via existing standards (bullet three), the more people will start using it and create new links to their own data sets. In the course of time, when more and more people put data collections and fragments together, the complete graph starts growing bigger and bigger. In Figure 5.7 we see the current overview of the linked data cloud. The figure shows all the currently available data sources and their interrelationships. Collectively, the data sets consist of over 4.7 billion RDF triples, which are interlinked by around 142 million RDF links (May 2009) [39].

\(^{11}\)http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
5.4. INSTANCE DATA INTEGRATION

Figure 5.7: Linking Open Data cloud diagram as of July 2009

To acquire information from this graph, the proper URI needs to dereferenced. Just like when you dereference a URL in the World Wide Web and obtain an HTML page, dereferencing a URI in the Semantic Web returns an RDF description, describing this particular URI. By dereferencing for example the URI http://dbpedia.org/resource/Kill_Bill at DBpedia\(^{12}\), we get an RDF description providing all the DBpedia triples known to describe this resource. However, inside the linked data graph, more triples can become available through new connections to other sources:

Subject: <http://dbpedia.org/resource/Kill_Bill>
Predicate: <http://www.w3.org/2002/07/owl#sameAs>
Object: <http://mpii.de/yago/resource/Kill_Bill>

This basically says that for this movie another source (in this case the Yago source) exists which contains an individual also describing this particular movie. Through this connection, a user/machine can navigate further to these sources retrieving even more RDF statements describing the movie resource and/or related concepts.

In the following subsections we describe different external sources which are used in iFanzy for metadata provisioning. Each of them has a particular value and focus, and can be used to describe or enrich television programs. The data from all these sources is integrated in the domain model presented in the previous section.

5.4.1 BBC Backstage

The British Broadcasting Corporation or BBC, which is the United Kingdom’s national television broadcaster, is known for its participation in innovative ideas and technologies. Via one of its R&D divisions called BBC Backstage\(^{13}\), BBC presents a pilot for open innovation which encourages people to make innovative applications. They deliver real-life data so that people can experiment with it. Because BBC was one of the major players in the TV-Anytime forum, they decided to

\(^{12}\)http://wiki.dbpedia.org/
\(^{13}\)http://backstage.bbc.co.uk/
publish all metadata of all their TV and radio channels on BBC Backstage in the TV-Anytime format. Ever since the 22th of June 2005, every day BBC Backstage puts a new file on the server containing the metadata of eight TV and eleven radio channels. Every file contains data of nine days starting from the present day.

Since the BBC channel metadata is already published in TV-Anytime format, there is no necessary transformation to align it with the iFanzy domain model. There is however a necessary translation from the available XML formatting to an RDF instance of the domain model. In the past, a lot of research was invested in defining ways to translate the contents of an XML file to other database formats like RDF or SQL. The majority of these approaches can be classified in two groups: the Structure-mapping approach and the Model-mapping approach, like first described in [263]. The structure-mapping approach basically tries to bring the specific XML structure (for example defined by a DTD) into the new format (a strategy described earlier in [165]). For us this would mean that we create an RDF file with exactly the same structure as the original XML file. The model-mapping approach on the other hand brings every arbitrary XML statement into the new format. Therefore, this approach can be applied on any given XML document. In other words, this approach is independent of the specific XML document’s structure. In practice, there are several implementations of these two techniques translating XML either to RDF or a relational format. In [233], Tous et al. follow the model-mapping approach, arguing that only this method can cover all XPath constructs, which is not possible with a structure-mapping strategy. Klein on the other hand, makes use of a structure-mapping approach achieving a generic translation from XML into RDF via an RDF Schema specification [143].

As previously mentioned, the ReDeFer toolkit contains a XML2RDF tool which can, given the ontology generated by the XSD2OWL tool, generate proper RDF [100]. The translation described by Garcia et al. also follows the structure-mapping approach and basically translates XML metadata instances to RDF instantiating the corresponding constructs in OWL. E.g. `xsd:elements` and `xsd:attributes` are translated to OWL:ObjectProperties for node to node relations and OWL:DatatypeProperties for node to value relations. During the translation, values are stored as simple types and RDF blank nodes (nodes in an RDF graph which are not identifiable by a URI and not literals) are introduced in the RDF model to serve as unknown nodes filling the holes in the graph. However, having blank nodes in the RDF graph, is not very practical. Most noticeably, the performance of querying such a database goes down and queries need to be constructed taking these blank nodes into account.

However, such a naive approach, introducing blank nodes in the RDF graph, is unacceptable. Moreover in [38], Bizer et al. discourage the use of blank nodes for Linked Data. Since it is impossible to set external RDF links to a blank node, merging data from different sources becomes much more difficult when blank nodes are used. In [229] on the other hand, Horst proves that entailment for RDF Schema (RDFS) is decidable, NP-complete, and in P if the target graph does not contain blank nodes.

Therefore, in order to retain the ability to uniquely identify every resource within the graph which was not identified in the XML file (and thus avoiding blank nodes), we developed a scheme to create unique URIs for every resource ourselves (translated from the XML description), complying with:

http://<namespace>/<classname>/<creationdate hash>_<counter>

The following (shortened) URI exemplifies for example a URI for an instance of the TV-Anytime class TVA:ScheduleEvent:

http://iFanzy.nl/Ontologies/TV-Anytime/ScheduleEvent/36b9abf...f2437bb_0

More recently also other researchers adopted similar approaches to avoid blank nodes in XML translations as shown in [46].

The library to enable the translation of a TV-Anytime XML description into an RDF representation, was achieved partially thanks to the Java library developed by BBC Backstage to read

14http://rhizomik.net/redefer/
5.4. INSTANCE DATA INTEGRATION

and write TV-Anytime descriptions\textsuperscript{15}. We extended this library, which could already read and write TV-Anytime descriptions in XML, by adding the ability to output valid RDF. In this way a straightforward means of acquiring RDF metadata from BBC Backstage was achieved.

5.4.2 XML-TV

XML-TV was already thoroughly introduced as a known audiovisual description scheme for television programs in Chapter 2. However, in Chapter 3 we saw that it did not qualify to serve as keystone of our central domain model given the requirements. Still, XML-TV is widely used on the Internet, and through it large quantities of television program metadata can be amassed. Therefore, it makes for a good source of metadata in our framework. However, the XML-TV schema is not TV-Anytime reconcilable and thus some transformations are required (the XML-TV schema, which is defined by the XML-TV DTD, can be found in Appendix A.1.1).

In contrast to the BBC Backstage metadata which was already described in the TV-Anytime format, the XML-TV data requires more steps to make it ‘understandable’ in our framework (i.e. translating from XML to RDF, transforming from XML-TV to TV-Anytime). After all, our main goal is to facilitate homogeneous access to a set of heterogeneous sources. This means that all the data from every source should be interpretable and support in answering user queries. To enable such transparent access to XML data sources, different strategies can be applied:

- Apply no transformations, and keep the XML as it is.
- Translate the XML source (both schema and instances) to an RDF(S) representation maintaining the original source’s structure.
- Translate the XML source (only the instances) directly to an RDF instantiation of the target model (in our case the enriched TV-Anytime schema).

In the first approach, no transformations are executed, and the XML remains how it is. However, when working with a central RDF(S)/OWL domain model which is used to answer among others user queries, there needs to be an interpretation of these queries. In other words, every incoming query needs to be altered and translated before being sent to the XML source. In [244] for example, Vdovjak et al. propose an architecture to integrate data from heterogeneous sources among which XML sources can appear. Centrally they defined an RDF(S) mediator which receives queries (adhering to the central conceptual model), fires modified queries to the individual sources and combines the obtained results afterwards. To be able to understand the RDF query, every XML source has a dedicated XML-RDF broker at its disposal which translates the incoming query to a correct XML equivalent. After execution, the XML query result is translated to an RDF representation which is then sent back to the mediator which integrates all obtained results. However, because of the on-demand character of this approach, there is a potentially large performance bottleneck. After all, on every request a set of sources is contacted, and for each one a whole pipeline of both query and result translation steps needs to be executed. Still, this approach is particularly interesting if the data is only available through an XML query endpoint. Moreover in [244], Vdovjak et al. advocate the benefit of working with online ‘live’ data because of the guaranteed freshness of the data. However, in our case with XML-TV, we obtain the data ourselves (new program data is retrieved every day) making us less sensitive for this issue.

In the second approach, the XML-TV schema and instances are translated to RDFS and RDF respectively, maintaining the XML-TV layout. The main advantage here is that you need to translate the retrieved XML sources to RDF only once (at retrieval time). Afterwards, the data is available for querying in RDF without further overhead. However, the XML-TV RDFS schema (which can be found in Appendix A.1.2) and our TV-Anytime based domain model differ in terms of semantics and modeling. This means that a domain model query cannot be answered by the XML-TV source. In such a case, every query $Q$ executed on model $M$ needs to be translated

\textsuperscript{15}The library/API is available at http://sourceforge.net/projects/tvanytimeapi/
into query $Q'$ which can be answered by model $M'$. This query translation step however can become complex when $M$ and $M'$ have different modeling approaches for similar concepts. In Figure 5.8 we see a small part of how both the XML-TV and TV-Anytime schema model program credits. In the XML-TV schema, the credits modeling is straightforward. At the top in Figure 5.8 we see a program which has a XMLTV:Credits class which in turn has a list of properties associated to guests, actors, presenters, directors, writers, etc. At the bottom of the figure we see the more complex model of TV-Anytime. Starting from the program’s basic content description, we see a property to a credits list (TVA:CreditsListType) which references a set of credit items (TVA:CreditsItemType). Every credit item has an associated property with the person’s name and the specific role this person fulfills in this program. The value of the role property is a class from a term hierarchy where we find among others the class actor, director, writer, etc. These two modeling approaches differ significantly and therefore require different queries to obtain the same information. Specifically the fact that the role of a person in the program is modeled in the name of a property in XML-TV and by means of a controlled term in TV-Anytime, complicates a potential query translation. However, there are several strategies described, to rewrite queries from one model to another, in the field of ontology alignment\textsuperscript{16}. In [141], Kalfoglou et al. show a good overview of the state of the art in this field.

The third option to allow transparent access to XML data sources is to translate the XML source (only the instances) directly to an RDF instantiation of the central conceptual model. An argument against this solution would be that this implies replication and local storage of the content, which could in time lead to an outdated copy. However, as previously mentioned, the retrieved XML-TV metadata is describing programs which will be broadcast in the near future. After being broadcast, the metadata becomes static and is not updated anymore. Arguments in favor of this approach includes firstly that it makes the extra query rewrite step (due to the difference in schema) superfluous thereby improving performance. Secondly, if a straightforward match between the two schemes can be made, we can easily generate a new TV-Anytime program instance and fill it with corresponding XML-TV values.

To verify whether such a straightforward match indeed can be made between XML-TV and TV-Anytime, let us take a look at the property correspondences. In Table 5.1, we see all XML-TV’s relevant properties (relevant in terms of describing a program) in the first column while in the second we see TV-Anytime’s counterparts expressed via a class and property combination. What really stands out in this list is the fact that all the most relevant properties are covered by TV-Anytime. The ones which are not (indicated by N/A) are in most of the times not really essential. For example the properties: XMLTV:videoplus, XMLTV:showview, XMLTV:vps-start and XMLTV:pdc-start describe different video recorder scheduling codes (showview and videoplus).

\textsuperscript{16}Also known as: Ontology mapping, integration, merging, articulation, etc.
### Table 5.1: Property overlap between XML-TV and TV-Anytime

<table>
<thead>
<tr>
<th>XML-TV Property</th>
<th>TV-Anytime equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel name</td>
<td>ServiceInformationType.Name</td>
</tr>
<tr>
<td>Channel URL</td>
<td>ServiceInformationType.ServiceUrl</td>
</tr>
<tr>
<td>Channel icon</td>
<td>ServiceInformationType.Logo</td>
</tr>
<tr>
<td>Subtitle type</td>
<td>N/A</td>
</tr>
<tr>
<td>Subtitle language</td>
<td>CaptionLanguageType.Language</td>
</tr>
<tr>
<td>Rating value</td>
<td>ParentalGuidanceType.MinimumAge</td>
</tr>
<tr>
<td>Rating icon</td>
<td>N/A</td>
</tr>
<tr>
<td>Rating system</td>
<td>N/A</td>
</tr>
<tr>
<td>Video aspect</td>
<td>VideoAttributeType.AspectRatio</td>
</tr>
<tr>
<td>Video color</td>
<td>VideoAttributeType.Color</td>
</tr>
<tr>
<td>Video present</td>
<td>BasicContentDescriptionType.Genre</td>
</tr>
<tr>
<td>Audio stereo</td>
<td>AudioAttributesType.NumOfChannels</td>
</tr>
<tr>
<td>Audio present</td>
<td>BasicContentDescriptionType.Genre</td>
</tr>
<tr>
<td>Credits Commentator</td>
<td>TVARoleCS.V32</td>
</tr>
<tr>
<td>Credits Presenter</td>
<td>TVARoleCS.V730</td>
</tr>
<tr>
<td>Credits Producer</td>
<td>RoleCS.Producer</td>
</tr>
<tr>
<td>Credits Guest</td>
<td>TVARoleCS.V97</td>
</tr>
<tr>
<td>Credits Adapter</td>
<td>TVARoleCS.V76</td>
</tr>
<tr>
<td>Credits Writer</td>
<td>RoleCS.SCRIPTWRITER</td>
</tr>
<tr>
<td>Credits Actor</td>
<td>RoleCS.Actor</td>
</tr>
<tr>
<td>Credits Director</td>
<td>RoleCS.Director</td>
</tr>
<tr>
<td>Premiere</td>
<td>ScheduleEvent.Type.FirstShowing</td>
</tr>
<tr>
<td>New</td>
<td>ScheduleEvent.Type.FirstShowing</td>
</tr>
<tr>
<td>Last chance</td>
<td>ScheduleEvent.Type.LastShowing</td>
</tr>
<tr>
<td>Previously shown</td>
<td>ScheduleEvent.Type.Repeat</td>
</tr>
<tr>
<td>Episode nr</td>
<td>EpisodeOf.Type.index</td>
</tr>
<tr>
<td>Country</td>
<td>BasicContentDescriptionType.ProductionLocation</td>
</tr>
<tr>
<td>URL</td>
<td>ProgramInformationType.OtherIdentifier</td>
</tr>
<tr>
<td>Length</td>
<td>ScheduleEvent.Type.PublishedDuration</td>
</tr>
<tr>
<td>Original language</td>
<td>ExtendedLanguage.Type.Language</td>
</tr>
<tr>
<td>Language</td>
<td>BasicContentDescriptionType.Language</td>
</tr>
<tr>
<td>VideoPlus code</td>
<td>N/A</td>
</tr>
<tr>
<td>Clump id</td>
<td>N/A</td>
</tr>
<tr>
<td>Showview code</td>
<td>N/A</td>
</tr>
<tr>
<td>Vps start</td>
<td>N/A</td>
</tr>
<tr>
<td>Pds start</td>
<td>N/A</td>
</tr>
<tr>
<td>Start</td>
<td>ScheduleEvent.Type.PublishedStartTime</td>
</tr>
<tr>
<td>Stop</td>
<td>ScheduleEvent.Type.PublishedEndTime</td>
</tr>
<tr>
<td>Title</td>
<td>BasicContentDescriptionType.Title</td>
</tr>
<tr>
<td>Sub-title</td>
<td>BasicContentDescriptionType.Title</td>
</tr>
<tr>
<td>Desc</td>
<td>BasicContentDescriptionTypeSynopsis</td>
</tr>
<tr>
<td>Date</td>
<td>BasicContentDescriptionType.ProductionDate</td>
</tr>
<tr>
<td>Category</td>
<td>BasicContentDescriptionType.Genre</td>
</tr>
<tr>
<td>Icon</td>
<td>BasicContentDescriptionType.MediaTitle</td>
</tr>
</tbody>
</table>

5.4. **INSTANCE DATA INTEGRATION**  

95
or program delivery control standards like PDC (Program Delivery Control) and VPS (Video Programming System). All four of these properties are less relevant for our specific application. Also the ratings system used by XML-TV is not really compatible with TV-Anytime (it misses the possibility to express different types of rating systems or icons), although pragmatical solutions can be thought of.

However, one issue remains. The XML-TV genres (from the category property) are not pre-defined in the XML-TV specification and thus arbitrarily chosen. After all, the value appearing in the category field matches the genre published on the source's Web site which was parsed by the grabber. The TV-Anytime instance on the other hand expects a term from a Classification Scheme (CS). To solve this issue we created a new XML-TV specific classification scheme listing all the XML-TV genres encountered so far. When a new genre is retrieved it is added to this CS as well. Then, we created a mapping between genres from the XML-TV CS and the original TV-Anytime CSs, by making use of the OWL construct `owl:sameAs`. Every time a new previously unknown XML-TV genre is added to the CS, the domain expert is signalled to add the appropriate mapping.

Alongside the BBC Backstage, XML-TV can be considered as a very important source. Its high value comes from the fact that it provides the complete broadcast program timetable forming the heart of the EPG for a multitude of countries and channels. Therefore, we call these (XML-TV and BBC Backstage) sources root sources. Their data is the necessary basis which can later be enriched by various other sources. Without it, other sources can not contribute much useful information at all. This and the previously given advantages make us conclude that it would be beneficial to have this metadata directly available as instance of the central domain model. The retrieved XML-TV data is translated upon retrieval time from XML directly to an extended TV-Anytime RDF instance.

### 5.4.3 IMDb

The Internet Movie Database\(^\text{17}\) (IMDb) is the most extensive movie data collection currently available online. According to IMDb\(^\text{18}\) they currently contain over 1.5 million titles among which there are roughly 450 thousand movies and 800 thousand TV programs. While IMDb initially started with movies, later IMDb was extended with TV-shows, games, etc. IMDb is an enormous potential source of metadata as it describes nearly every imaginable movie feature like soundtracks, awards, filming locations, etc. of almost every movie known to exist.

IMDb has a program where people can license its data\(^\text{19}\) and use within their own setting. This data can also be acquired for free if the usage is of a non-commercial nature (see the license), like for example in research. The freely available IMDb database is split in 49 files either listing objects like movies or listing connections between objects like actors (from actor to movies he or she played in), genres (from movies to their genre), composers, writers, costume designers, etc. Although these files are well structured, the format is a simple text file which makes it hard to search through. The structure is shown at the beginning of each file and differs for every one them. We can classify these files in:

- **Cast Lists**: actors and actresses
- **Crew Lists**: composers, directors, costume designers, producers, writers, etc.
- **Person Detail Lists**: biographies, etc.
- **Movie Detail Lists**: certificates, color information, countries, genres, goofs, keywords, languages, literature, movie links, plots, quotes, etc.
- **Company Lists**: distributors, productions, special effects, etc.

---

\(^{17}\)http://www.imdb.com/

\(^{18}\)http://www.imdb.com/database_statistics

\(^{19}\)http://www.imdb.com/interfaces
To uniquely identify a movie, IMDb uses the name of the movie in combination with the year of launch. However, if more than one movie with the same name was launched in the same year, a Roman number is used to guarantee uniqueness. For a person, it often happens that he or she shares his or her name with his or her father or mother. Also in those cases a roman number is used to extend the name and guarantee uniqueness. In Figure 5.9, we see a (shortened) entry of "Scarlett Johansson" in the actresses file. In this list we see all her appearances (both in movies, series, public, etc.). In this particular file the syntax includes round brackets for the year, squared brackets for the name of the character he or she portrayed and diamond brackets for the position in the cast list if applicable. For more information on this translation from text to RDF, we would like to refer to the thesis of one of our master students [216].

![Table: IMDb Actresses Entry]

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
<th>Role</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Girl with a Pearl Earring</td>
<td>2003</td>
<td>[Griet] 2</td>
<td></td>
</tr>
<tr>
<td>Gomorra</td>
<td>2008</td>
<td>[Herself]</td>
<td></td>
</tr>
<tr>
<td>Lost in Translation</td>
<td>2003</td>
<td>[Charlotte] 1</td>
<td></td>
</tr>
<tr>
<td>The Black Dahlia</td>
<td>2006</td>
<td>[Kay Lake] 2</td>
<td></td>
</tr>
<tr>
<td>The Other Boleyn Girl</td>
<td>2008</td>
<td>[Mary Boleyn] 2</td>
<td></td>
</tr>
<tr>
<td>The Prestige</td>
<td>2006</td>
<td>[Olivia Wenscombe] 6</td>
<td></td>
</tr>
<tr>
<td>&quot;The Tonight Show with Jay Leno&quot;</td>
<td>1992</td>
<td>[Herself]</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.9: Example of an entry in the IMDb Actresses file

Obviously, this data is not really usable in its present formatting. The structure is not using any standardized format and searching would be limited to an inefficient text search without any form of semantic relations to utilize. A transformation of this vast information source into a format which would make it semantically richer and allow it to be aligned with our central domain model is the way to unlock this immense data repository. Just like in the case with XML-TV, there are different ways to effectuate this translation. The most obvious choice is again to either translate the source data to an RDF(S) representation maintaining the original IMDb file structure or to translate the source directly to an RDF instantiation of the enriched TV-Anytime RDF(S)/OWL schema. While we argued in favor for the latter in the case of XML-TV, the IMDb introduces different characteristics. The main argument to translate the XML-TV data as a direct TV-Anytime instance arose from the fact that all the most relevant XML-TV properties could easily be converted to a TV-Anytime counterpart. However, the IMDb source has a large number of properties which have no appropriate counterpart in TV-Anytime. Therefore, a translation to TV-Anytime would induce a relative large loss of information.

In Appendix A.2 we see the RDFS schemas we constructed (based on the available structure in the text files) to contain the IMDb data. For reasons of convenience and overview, we have split the schema representation in seven parts. In Appendix A.2.1 we see the part of the schema describing an IMDb title (which comprises both movies, series, episodes, etc.). An IMDb title is the central concept. It has, besides a whole list of literal properties, also relations to photos, trailers, articles, books, other titles, etc. Relations between different titles are among others IMDb:referencedIn, IMDb:followedBy, IMDb:spinOffFrom, IMDb:remakeOf, etc. In Appendix A.2.2 we see the schema modeling a person in IMDb. As seen in the schema, a person can have one or more roles in one or more IMDb titles. Different roles like IMDb:Actor, IMDb:Director, IMDb:ProductionDesigner, etc. are defined, and subclasses of the class IMDb:Role. The model also shows relations between persons like e.g. IMDb:sonOf, IMDb:brotherOf, etc. In Appendix A.2.3 and A.2.4 we see the schema of the literature and company relations respectively. The former describes all the appearances of both persons and titles in different books and/or articles (published in a specific magazine), as well as books or stories on which a specific title was based on. The latter describes all the companies which were involved in the production and distribution of a title. Appendix A.2.6 and A.2.7 describe associated photos and trailers respectively in more detail. Finally, in Appendix A.2.8 we see some extra properties which are available in case a particular title is an episode of a series (e.g. IMDb:season and IMDb:episodeNr). In most of these IMDb schemas
we see properties like IMDB:filmingLocation, IMDB:birthPlace, IMDB:companyLocation, etc. referring to geographical locations modeled as class IMDB:Location. IMDB uses a huge hierarchical locations tree\(^{20}\) which is used when making references to geographical locations. These locations include besides regular countries, provinces and cities, also seas, oceans, marine ships, the corner of two streets, houses, etc. The IMDB set contains 48,612 unique locations divided over 217 top level elements (e.g. countries)\(^{21}\). In Figure 5.10 we see two location entries from the IMDb hierarchy. The second line for example shows four levels, namely a street, a city, a province and a country. Generally, to model such structures, part-whole relations (like the transitive partOf and hasPart properties) of the classical mereology theory are used \cite{109}. This structure can easily be used to say for example that the city Rotterdam is a part of the province Zuid-Holland. The structure of the schema can be seen in Appendix A.2.5. In order to connect all top level elements in one big tree, we added a class IMDB:World which serves as the general parent of all 217 of them.

Rancho Palos Verdes, California, USA
Grote Kerkplein, Rotterdam, Zuid-Holland, Netherlands

Figure 5.10: Example of an entry in the IMdb locations hierarchy

Following the Semantic Web philosophy and the effort to create new links between different data sources improving searching strategies, we made a connection between the GeoNames source and the IMDB locations tree. Because of the hierarchical structure of the tree, we devised a strategy making connections to GeoNames classes through the OWL:sameAs property. First, we made a straightforward connection between every country in the tree and the respective country in the GeoNames source. This first process was automated but also manually verified at the end to ensure correctness at the start of the linking process. Secondly, in a Breadth-first fashion, for every country we go through all its direct children and try to match those to classes in GeoNames which have a GEO:parentFeature (including transitivity) to this particular country. Lastly, we repeat this step for all children of every child in the previous step. This new connection allows for example to search for movies which were filmed in the same geographical area as a particular program.

Further, in all the IMDB RDFS schemas, all time related properties, expressing for example durations and instants in time, are modeled by instances of the OWL-Time ontology during this translation step. Just like XML-TV, IMDB also contains a (limited) set of genres, and similarly, we translated this list into a new CS genre listing. The translation of the data (the 49 text files) to RDF is done once (for more information see \cite{216}). Afterwards, only files which are updated (indicated by IMDB) need to be re-parsed to keep our repository up-to-date.

As previously mentioned, the XML-TV and BBC Backstage sources can be regarded as root sources. They provide the initial program schedules necessary to show program listings in for example regular paper EPGs. IMDB on the other hand, can be regarded as an enrichment source. It is solely used to enrich the description of programs being broadcast (or available via a Video-On-Demand service) rendering it useless in this setting without an extensive root source.

In Figure 5.1 we saw the RDF representation of the program $P$, describing the movie “The Alzheimer Affair” directed by “Erik Van Looy”. In Figure 5.11 we see $P$ again, but this time enriched by the IMDB RDF instance describing that exact movie. The enrichment is established on instance level by utilizing the OWL:sameAs construct. With this property, the correct IMDB instance is now regarded as the same individual as the original program $P$. Through this link, $P$ obtained six IMDB specific properties like e.g. IMDB:hasAkaTitle (showing here the Belgian title), IMDB:hasRating (providing the IMDB rating), IMDB:filmingLocation (one of the filming locations), IMDB:genre (the genre which is also connected to the TV-Anytime genre classification), etc.

Matching a program to a movie in the IMDB source is facilitated by means of a metadata comparison. To start, we look at the title in combination with the production year. After all, we see properties like IMDB:filmingLocation, IMDB:birthPlace, IMDB:companyLocation, etc. referring to geographical locations modeled as class IMDB:Location. IMDB uses a huge hierarchical locations tree\(^{20}\) which is used when making references to geographical locations. These locations include besides regular countries, provinces and cities, also seas, oceans, marine ships, the corner of two streets, houses, etc. The IMDB set contains 48,612 unique locations divided over 217 top level elements (e.g. countries)\(^{21}\). In Figure 5.10 we see two location entries from the IMDb hierarchy. The second line for example shows four levels, namely a street, a city, a province and a country. Generally, to model such structures, part-whole relations (like the transitive partOf and hasPart properties) of the classical mereology theory are used \cite{109}. This structure can easily be used to say for example that the city Rotterdam is a part of the province Zuid-Holland. The structure of the schema can be seen in Appendix A.2.5. In order to connect all top level elements in one big tree, we added a class IMDB:World which serves as the general parent of all 217 of them.

Rancho Palos Verdes, California, USA
Grote Kerkplein, Rotterdam, Zuid-Holland, Netherlands

Figure 5.10: Example of an entry in the IMdb locations hierarchy

Following the Semantic Web philosophy and the effort to create new links between different data sources improving searching strategies, we made a connection between the GeoNames source and the IMDB locations tree. Because of the hierarchical structure of the tree, we devised a strategy making connections to GeoNames classes through the OWL:sameAs property. First, we made a straightforward connection between every country in the tree and the respective country in the GeoNames source. This first process was automated but also manually verified at the end to ensure correctness at the start of the linking process. Secondly, in a Breadth-first fashion, for every country we go through all its direct children and try to match those to classes in GeoNames which have a GEO:parentFeature (including transitivity) to this particular country. Lastly, we repeat this step for all children of every child in the previous step. This new connection allows for example to search for movies which were filmed in the same geographical area as a particular program.

Further, in all the IMDB RDFS schemas, all time related properties, expressing for example durations and instants in time, are modeled by instances of the OWL-Time ontology during this translation step. Just like XML-TV, IMDB also contains a (limited) set of genres, and similarly, we translated this list into a new CS genre listing. The translation of the data (the 49 text files) to RDF is done once (for more information see \cite{216}). Afterwards, only files which are updated (indicated by IMDB) need to be re-parsed to keep our repository up-to-date.

As previously mentioned, the XML-TV and BBC Backstage sources can be regarded as root sources. They provide the initial program schedules necessary to show program listings in for example regular paper EPGs. IMDB on the other hand, can be regarded as an enrichment source. It is solely used to enrich the description of programs being broadcast (or available via a Video-On-Demand service) rendering it useless in this setting without an extensive root source.

In Figure 5.1 we saw the RDF representation of the program $P$, describing the movie “The Alzheimer Affair” directed by “Erik Van Looy”. In Figure 5.11 we see $P$ again, but this time enriched by the IMDB RDF instance describing that exact movie. The enrichment is established on instance level by utilizing the OWL:sameAs construct. With this property, the correct IMDB instance is now regarded as the same individual as the original program $P$. Through this link, $P$ obtained six IMDB specific properties like e.g. IMDB:hasAkaTitle (showing here the Belgian title), IMDB:hasRating (providing the IMDB rating), IMDB:filmingLocation (one of the filming locations), IMDB:genre (the genre which is also connected to the TV-Anytime genre classification), etc.

Matching a program to a movie in the IMDB source is facilitated by means of a metadata comparison. To start, we look at the title in combination with the production year. After all,

\(^{20}\)http://www.imdb.com/LocationTree

\(^{21}\)Numbers from 2007, in 2009 there were 275
Figure 5.11: Movie instance enriched by IMDb
there are not many different programs with the same title produced in the same year. Do note that some countries translate the title of the programs they broadcast (e.g. Italy, France, Spain, etc.). In such a case, we utilize the IMDB:hasAkaTitle property which lists alternative titles for different countries. For example, the movie “Il buono, il brutto, il cattivo” has a UK name “The Good, the Bad and the Ugly”. In case this first search, based on title and production year, lists more than one result, we also consider the values for director and actors to verify the correct match. Another approach is presented in [72], where Keijzer et al. apply a probabilistic approach at the integration problem, by applying simple and generic knowledge rules. Matches with a high certainty can then automatically be added. They also showcase their approach with an IMDb scenario.

5.4.4 VideoDetective

Introducing real content in research projects is always difficult, and particularly if the content is copyright sensitive. Unfortunately, this is not different in iFanzy, because due to its concept of personalizing the television watching experience, having visual content could contribute a lot to the use case. However, having this luxury would only be possible when working together closely with a content provider. Yet even without, in iFanzy we currently visualize content by integrating the Videodetective22 trailer source. This source not only enables us to show trailers for television programs (with a clear focus on movies though), it also shows that integration of external data sources is a straightforward thing to do.

```xml
<!DOCTYPE rdf:RDF [<!ENTITY tva "http://<our namespace>/">
   <!ENTITY cs "http://<classifications>/">]>
<rdf:RDF>
  <tva:ProgramInformationType rdf:about="&tva;a12vcnd9asbf5122c0_0">
    <tva:BasicDescription rdf:resource="&tva;5df6dmgk53fng9a1r_5"/>
  </tva:ProgramInformationType>

  <tva:BasicContentDescriptionType rdf:about="&tva;5df6dmgk53fng9a1r_5">
    <tva:Genre rdf:resource="&cs;genre/3.5.7"/>
    <tva:Title>Le fabuleux destin d’Amélie Poulain</tva:hasTitle/>
    <tva:RelatedMaterial rdf:resource="&tva;1rh3w0sdnfk14du6n6_0"/>
  </tva:BasicContentDescriptionType>

  <tva:RelatedMaterialType rdf:about="&tva;1rh3w0sdnfk14du6n6_0">
    <tva:HowRelated rdf:resource="&cs;HowRelated/1"/>
    <tva:SourceMediaLocator rdf:resource="&tva;u0lv21gfdg54d3w6g8_7"/>
  </tva:RelatedMaterialType>

  <tva:MediaLocatorType rdf:about="&tva;u0lv21gfg54d3w6g8_7">
  </tva:MediaLocatorType>
</rdf:RDF>
```

Figure 5.12: Program metadata including a VideoDetective trailer

VideoDetective claims to be the biggest provider of trailers (with an estimation of 70.000 trailers). It contains trailers from theatrical releases, TV and DVD releases. VideoDetective can be regarded as one of the only resources offering free (non-commercial) access (after registration) to a vast trailer database in return for watching some advertisements shown in the player. Given some search terms, the open interface returns an XML document containing a list of results each.
5.4. INSTANCE DATA INTEGRATION

showing the metadata describing every single trailer, matching the search terms. This metadata contains descriptive information like title, actors, release date, director, etc. next to the link pointing to the trailer itself. Often, a query returns a list showing many different results. Having this metadata, the matching strategy is almost identical to the one applied when searching for matches with IMDb.

To embed or group different content assets like pictures, soundtracks, making-of’s, trailers, etc. in a TV-Anytime program description, two separate methods can be applied. Firstly, they encourage the generation of packages (TV-Anytime Phase II) where different assets can be grouped together in one thematic package like e.g. a Lord Of The Rings package. While a trailer can also be described as a part (asset) of a package, it introduces quite an overhead in terms of metadata. The second TV-Anytime provided method works by defining related material. Every program can have a list of items within its own metadata description which are considered related to the program. Every one of those items has some properties among which there are two important ones: the location of the item and the type of relatedness. The value of this relatedness property (TVA:HowRelated) is again a term from a CS, namely the HowRelatedCS. Among the terms of this classification scheme we find for example adverts, “more information”, “making-of”, etc., but also the term ‘Trailer’. By means of this term we can now easily connect the trailer as an asset of any TV-Anytime described program as shown in Figure 5.12 (due to space constraints the URIs are shortened). Just like IMDb, VideoDetective is considered an enrichment source, here enabling the integration of video trailers.

5.4.5 DBpedia

Since its launch in 2001, Wikipedia\(^\text{23}\) has become one of the largest publicly available multilingual encyclopedias online. Ever since its launch, Wikipedia has been improved and corrected by volunteers and fanatics, giving it a strong Web 2.0 (community) character [56]. Due to its popularity at one hand and its immense information availability on the other (as of September 2009 thirteen million articles among which three million in the English Wikipedia), Wikipedia drew the attention of researchers. Initially Denoyer et al. proposed in [76] a set of XML collections based on Wikipedia. Later, also the Semantic Web community started utilizing the data. In [142] for example, Kinzler extracted topic-maps and constructed an ontology. Later, researchers started to make RDF versions of the data set [226] and category links [62]. However, the most successful approach to an RDF version of Wikipedia data to this date, is DBpedia\(^\text{24}\).

DBpedia has evolved from research trying to exploit the strong points of the Wikipedia platform by adding more structure and semantics [19]. The data is elicited from information which resides within a Wikipedia page, more concretely within the structured infobox. As mentioned on their Web site (as of September 2009): *The DBpedia knowledge base currently describes more than 2.6 million things, including at least 213,000 persons, 328,000 places, 57,000 music albums, 36,000 films, 20,000 companies. The knowledge base consists of 274 million pieces of information (RDF triples). It features labels and short abstracts for these things in 30 different languages; 609,000 links to images and 3,150,000 links to external Web pages; 4,878,100 external links into other RDF data sets, 415,000 Wikipedia categories, and 75,000 Yago categories.* While this is a vast source of RDF data, the majority of the information has no relation with the television domain at all. However, we are mainly interested in the enrichment of program descriptions with background information from external sources. Via the DBpedia Snorql query browser\(^\text{25}\) and some simple SPARQL (SPARQL Protocol and RDF Query Language) queries we could estimate the number of potentially relevant classes with respect to the TV domain, as shown in Table 5.2. Via this public SPARQL endpoint, we can discover extra information about programs, people, etc. and connect these resources to our own TV-Anytime instance descriptions via the OWL:sameAs property. Nowadays, DBpedia is even used by big companies like BBC\(^\text{26}\).

\(^{23}\)http://www.wikipedia.org/

\(^{24}\)http://wiki.dbpedia.org/

\(^{25}\)http://dbpedia.org/snorql/

To be able to exploit the DBpedia data, the relevant parts were added to our data graph. This was necessary because the SPARQL endpoint alone is not enough to achieve our goals. Indeed, whenever we need data about a concept or search the DBpedia space, we can use the endpoint, however, a reasoner can only perform on a given set of triples. Therefore, if two concepts are connected via the OWL:sameAs property and one of those two resources resides in a separate online repository, the reasoner will not automatically retrieve the necessary data. Until such functionality becomes the new state-of-the-art, necessary data needs to be stored locally as well.

5.5 Overview

In the previous two sections we have discussed the integration of TV program data from various heterogeneous sources. These integration strategies (especially) proved valuable when the TV program’s annotation is limited. In this section, we show an example of a program that went through this enrichment pipeline, and illustrate the complete structure.
To start, in Figure 5.13 we see a small but relevant part of the TV-Anytime RDFS schema extended with some new iFanzy classes as described in Section 5.4 to describe a program. We see the newly introduced IFZ:Location class to describe geographical concepts, links to the two OWL-Time classes to describe time instants and duration descriptions and the SKOS enriched IFZ:GenreType class. More interesting is the example of a simplified program instance (after enrichment), as shown in Figure 5.14. This program, modeling the movie “The Godfather”, exemplifies the different integration steps we discussed in this chapter (do note that due to space constraints some instance names are shortened). Starting from the top to the bottom, we see the class IFZ:godfather1972 as the central class modeling this program. Due to enrichment by sources like IMDb and DBpedia, we see the instances DBpedia:The Godfather and IMDb:tt0068646, which model the exact same instance, connected via the OWL:sameAs property. We also see the same relation between the DBpedia concept and the Yago representation, which exists within the DBpedia’s description. We also see the link IFZ:followedBy to the second installment of the trilogy. The program is described by the basic content description class IFZ:description. This description has a synopsis, a title and one or more keywords. In Figure 5.14 we see the keyword ‘mafia’ modeled by the instance TVA:keyword. This instance in turn is connected to the instance wn20instances:word-mafia which models the exact same word in the WordNet ontology. Both are connected via the OWL:sameAs property. The creation coordinates property shows locations where this program was filmed (or created). The instance IFZ:coordinate_1 has a link to the iFanzy instance IFZ:location_2 modeling the geographical location, which in turn is marked to be the same resource as Geo:5128581 and TGN:7007567 both modeling the filming location “New York City”. Next, is a link to the IFZ:relatedMaterial_1 instance which links via a media locator resource to the correct VideoDetective trailer for “The Godfather”. The movie’s production date is modeled via the IFZ:timeType_1 instance which points to the original string (“1972-03-15”) via the original property TVA:TimePoint as well as to a time instant description IFZ:dateTimeDesc via the property IFZ:TimePoint which models the same information as an instance of the OWL-Time ontology class. Next to a genre and a duration description (which is also an instance of an OWL-Time ontology class), we also see the credits list instance IFZ:creditsList_1. This list consists of one or more credits items (here we only show one: IFZ:creditsItem_1), which shows the name of the person, the role he or she had in this program and in case of acting the character’s name. Lastly, we see the production location of the program (in this case ‘USA’), which is again modeled by a location instance and again connected to the TGN and GeoNames sources via the OWL:sameAs property.

<table>
<thead>
<tr>
<th>Knowledge Source</th>
<th>Nr. of triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV-Anytime Genres</td>
<td>±3.971</td>
</tr>
<tr>
<td>OWL-Time</td>
<td>±1.547</td>
</tr>
<tr>
<td>WordNet</td>
<td>±2.579.206</td>
</tr>
<tr>
<td>TGN</td>
<td>±425.517</td>
</tr>
<tr>
<td>GeoNames</td>
<td>±93.896.732</td>
</tr>
<tr>
<td><strong>TV Data Source</strong></td>
<td><strong>Nr. of triples</strong></td>
</tr>
<tr>
<td>BBC Backstage</td>
<td>±91.447(weekly with 3 channels)</td>
</tr>
<tr>
<td>XML-TV</td>
<td>±1.278.718 (weekly with 12 channels)</td>
</tr>
<tr>
<td>IMDb</td>
<td>±65.994.643</td>
</tr>
<tr>
<td>VideoDetective</td>
<td>±350.000(taking 5 triple per trailer)</td>
</tr>
<tr>
<td>DBpedia</td>
<td>±274.000.000</td>
</tr>
</tbody>
</table>

Table 5.3: Triple sizes of the most important sources used

To conclude this overview, we include some statistics with respect to the sizes of different sources. In Table 5.3 we see an overview of sizes (in number of triples) of all the used sources including both knowledge and data sources. Numbers partially come from direct measurements\textsuperscript{27}

\textsuperscript{27}Measurements were done in October 2007
Figure 5.14: Example instance exemplifying sources integration
5.6 Conclusion

Figure 5.15: iFanzy data sources and connections

and partially (in case of TGN) from [210]. In case of VideoDetective, we take that it can maximally contribute with one triple per trailer. For root sources like BBC Backstage and XML-TV, the number of triples varies every week and of course also on the number of channels retrieved through this source. Lastly, in Figure 5.15 we see the data sources and connections used in iFanzy depicted as a small cloud. The four sources at the bottom and depicted in orange, are also major sources included in the Linked Data cloud which could be seen in Figure 5.7. The five sources at the top and shown in blue, are sources used in iFanzy which were not yet available in the Linked Data cloud.

5.6 Conclusion

In this chapter, we discussed the data integration component in the approach proposed in Chapter 4. By integrating data from external and potentially heterogeneous sources, we were able to enrich the metadata of television programs to a large extent. Since every source has its own focus and perspective, the added value after integration and combination of the data, through Semantic Web techniques, became even larger than when interpreting the data of each source separately. This holistic approach combines data in such a way that it becomes usable for rich information retrieval and personalization of television programs.

Section 5.3 of this chapter described how schematic enrichments help in describing television programs. The approach included temporal and geographical sources as well as lexical and hierarchical improvements. These additions enable more fine-grained search patterns like places which are part of a bigger whole, genres which are related to others, time restrictions on the level of minutes, synonym relations, etc. Section 5.4 on the other hand, provided an overview of various metadata sources providing instances and descriptions specifically for the enrichment of program metadata. Different enrichment sources available online like DBpedia, IMDb, VideoDetective, etc. have huge collections of data which are already impressive individually but even more imposing when combined. They are able to both complement as well as fill in missing information of the available root sources like BBC Backstage and XML-TV. Further, the example at the end of the section showed a glimpse of how such an enriched program would look after the integration of other sources’ data.

The integration of heterogeneous sources proposed in this chapter proved that the extended version of the TV-Anytime specification is able to connect all different data elements in an orderly way without breaking backward compatibility. Although sometimes a transformation was necessary, all chunks of data and data assets were combined into one graph adhering to one consistent schema. This chapter therefore provided an answer to research question 4 (How can we integrate large heterogeneous data sources into one consistent and semantically rich data model?).
5.7 Related Work

One of the key points of Semantic Web is the integration of different data sources across various domains. In this section, we first take a look at different state-of-the-art projects which also benefit from the integration of ontological knowledge, vocabularies and thesauri but also of instances. Afterwards, we provide an overview of some ontology alignment and integration techniques.

5.7.1 State-of-the-art applications

The main objective of the MultimediaN E-Culture project [210] is to demonstrate how novel Semantic Web and presentation technologies can be deployed to provide better indexing and search support within large virtual collections of cultural-heritage resources. Similarly in the CHIP project [16], where Aroyo et al. present the same idea but focussed on digital museum collections. Unlike E-Culture, in CHIP, the user can build a profile, by rating art works, which in turn is used to generate recommendations. Based on this profile, CHIP is also able to generate a personalized museum tour. Both E-Culture and CHIP make use of the three Getty vocabularies28 i.e. the Art and Architecture Thesaurus (AAT), Union List of Artists Names (ULAN) and the Thesaurus of Geographical Names (TGN). E-Culture further also deploys WordNet and Dublin Core for search and annotation purposes. CHIP on the other hand uses the subject classification Iconclass29. Both utilize among others the ARIA collection30 of the Rijksmuseum in Amsterdam containing images of some 750 master pieces. Various connections, between ARIA terminology to the AAT, ULAN, TGN and IconClass concepts, like for example the concepts for places in ARIA refer to location terms in TGN, styles in AAT are linked to artists in ULAN, birth places of artists in ULAN refer to location terms in TGN, etc., are made. Related is the Cultural Heritage Interface (CHI) project [11], where the goal is to bring cultural heritage collections to the general public. To make these collections available, a CHI ontology was developed enriched which other knowledge sources like WordNet, SKOS and GeoNet for geographical information.

Various new innovative concepts with respect to searching, navigation, recommendation, etc. are nowadays making more and more use of Semantic Web techniques. Revyu is a live, publicly accessible reviewing and rating Web application, designed to be usable by humans whilst transparently generating machine-readable RDF metadata for the Semantic Web, based on their input [117]. Similarly to iFanzy, Revyu uses ontologies like FOAF [49] and Tag[180], and URIs from Yago, DBpedia, etc. connected via the OWL:sameAs property. GroupMe! is an application which allows users to group and arrange multimedia Web resources they are interested in [3]. The GroupMe! approach provides the possibility to overcome the gap between Web 2.0 and the Semantic Web by using well known vocabularies like FOAF, RSS31, Dublin Core and a custom GroupMe! vocabulary. paggr is an interactive application that simplifies the organization and integration of distributed Web information [185]. In a few simple steps, users can create personalized, “smart data” portals from a variety of sources and formats such as RDF, remote SPARQL endpoints, microformats, RSS, Atom, RDFa (RDF in attributes), or selected APIs, providing novel ways to manage and repurpose information on the Web. Sig.ma is both a service and an end-user application to browse and perform tasks leveraging data coming from dozens of distributed and unrelated sources on the Web of Data [236]. Via a keyword search, Sig.ma is able to find various pieces of RDF data, which it then joins to present a clear RDF data representation of the matching subject.

In [234], Tsinaraki et al. describe a systematic methodology for extending the audiovisual content description standards (MPEG-7 and TV-Anytime) with domain-specific knowledge descriptions expressed in OWL. Their Domain-Specific Multimedia Indexing, Retrieval and Filtering (DS-MIRF) framework produces consistent MPEG-7 and TV-Anytime compliant semantic descriptions for the audiovisual content by means of semantic indexing. This indexing process is guided by appropriate domain-specific ontologies, which permit more accurate descriptions of the

28http://www.getty.edu/research/conducting-research/vocabularies
29http://www.iconclass.nl/libertas/ic?style=index.xsl
30http://rijksmuseum.nl/aria/
31http://web.resource.org/rss/1.0/
5.7. RELATED WORK

concepts of each application domain than the general-purpose concepts described by the standards themselves. The domain-specific ontologies are based on the Semantic Part of the MPEG-7 Multimedia Description Schemes (MDS). In other words, they have defined a core OWL ontology, which fully covers the Semantic Part of the MPEG-7 MDS, and then defined the domain-specific ontologies in OWL as extensions of the core ontology to enhance semantics. In principle, this is similar to iFanzy in a sense that we also enrich program metadata (in an OWL version of the TV-Anytime/MPEG-7 format) by means of external sources. However, in iFanzy we deal with data from already existing external sources, and adapt to their structure. In [234], every external domain-ontology must be defined by using the MPEG-7 syntax and constructs.

HealthFinland is a national semantic publishing system for providing Finnish citizens with reliable, up-to-date information about health [132]. It is accessible via an intelligent semantic portal aggregating and presenting the contents coming from several distributed health organizations. A centralized service of health ontologies together with tools helps to support the content structuring. Besides using three medical vocabularies (MeSH\textsuperscript{32}, UMLS\textsuperscript{33} and SNOMED CT\textsuperscript{34}), the HealthFinland project utilizes knowledge from Dublin Core, SKOS, the Finnish General Upper Ontology (YSO) (generated from the General Finnish Thesaurus (YSA)) and The European Multilingual Thesaurus on Health Promotion (HPMULTI).

An even broader overview of Semantic Web related applications, can be found at [231].

5.7.2 Alignment and integration techniques

In iFanzy, most of the relations and links between different ontologies, vocabularies and thesauri have been realized by either using the OWL:sameAs property, by defining a RDFS:subClassOf inheritance relation or by the intervening hand of a domain expert. However, also other ontology mapping/alignment techniques exist. In fact, many different approaches exist, which usually can be classified in different categories like e.g. translators, mediators, methods, etc. Therefore also many overviews exist. In [141], Kalfoglou et al. made a very extensive overview of 35 different approaches, providing insights on the pragmatics of ontology mapping. In [213], Shvaiko et al. present a new classification (and taxonomy) of schema-based matching techniques that builds on top of state-of-the-art in both schema and ontology matching. The survey of [248] reviews the use of ontologies for the integration of heterogeneous information sources. They concentrate on ontology-based information integration and discuss general matching approaches. In [240], various ontology alignment techniques are evaluated.

Considering the large number of different approaches, we limit ourselves here by only listing the most related ones, also dealing with Semantic Web data.

Hera is a methodology that supports the design and engineering of Web Information Systems (WIS), by using Semantic Web techniques. It is a model-driven methodology that distinguishes three parts in the design: integration, data retrieval, and presentation generation [243]. The integration part is responsible to retrieve data from different heterogeneous sources and align it with the Conceptual Model (CM), modeling the domain of the WIS. To facilitate these alignments, they rely on a domain expert to articulate CM concepts in the semantic language of sources. Such an articulation is a specification to describe actual links between the CM and the source ontologies. Consequently, every Hera query to the CM must be translated on the fly to a set of source queries, and afterwards their respective results integrated, by means of these articulations.

Due to the abstraction level of Semantic Web concepts and complexity of the available tools, it often remains too elusive for casual users. Potluck is a Web user interface that lets casual users, those without programming skills and data modeling expertise, merge and repurpose heterogeneous Semantic Web data [131]. It lets users merge, navigate, visualize, and clean up data all at the same time, using direct visual manipulation. Users who find for example multiple useful sources of semantic Web data, can instantly merge them into one single blob in their Web browser. In other words, the end-user unconsciously helps in aligning ontologies just by using the Web site.

\textsuperscript{32}http://www.nlm.nih.gov/mesh/
\textsuperscript{33}http://umlsinfo.nlm.nih.gov
\textsuperscript{34}http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html
Besides having explicit expert’s alignments or depending on end-users to do the work, other research focusses on algorithms trying to (semi-)automatically discover correct mappings [197]. PROMPT is an algorithm that provides a semi-automatic approach to ontology merging and alignment [186]. PROMPT performs some tasks automatically and in the other case, where the user’s intervention is required, PROMPT guides the user in performing particular tasks. PROMPT also determines possible inconsistencies in the state of the ontology, which can result from the user’s actions, and suggests ways to remedy these inconsistencies. GLUE is a system that employs machine learning techniques to find mappings between two ontologies. Given two ontologies, for each concept in one ontology GLUE finds the most similar concept in the other ontology [78]. GLUE makes heavy use of the fact that many data instances are associated with the ontologies they are trying to match. GLUE further employs several learners like for example nearest-neighbor classification, regression, Naive Bayes, etc. and was able to accurately match 66-97% of the nodes on several real-world domain ontologies.

Similarly to OWL mapping properties like OWL:sameAs and OWL:equivalentClass, SKOS also defines a set of mapping properties. SKOS goes in this regard beyond the expressivity of OWL by introducing properties like SKOS:closeMatch, SKOS:narrowMatch, SKOS:relatedMatch, etc. However, these properties are limited to SKOS hierarchies only by means of their domain and range properties which need to be of type SKOS:Concept.
Chapter 6

User Modeling

A completely personalized world would be an interesting place to live in. Every single person would experience a different personal reality as a unique instance of the world's ontology. Every person is different. Different dreams, habits, behavior and background, but also different language, preferences, ideas and purpose. These differences between individuals are exactly what makes for example shopping a painstakingly long process. After all, shops need to make sure they have something for everyone, which by default makes it more difficult to find what they have for you.

In a personalized world, every shop offers you exactly what you need in terms of look and feel, size, price, etc. while all the music, movies, advertisements played around you fit your personal liking and style. When you come home, the automatic shopping delivery brought you the food you like to eat, the lights dim to your favorite level and the television starts playing your most-beloved TV program. Although purely technically, this would all be possible with current technologies, pragmatically and ethically however there are still issues to resolve. While we leave the ethical and privacy concerns to other knowledgeable people, a general practical implementation of such a system would also soon run into some large hurdles.

The main difficulty comes from the fact that such a system would require an extremely accurate, up-to-date and computer-understandable representation of the user. However, such a representation or model can only be constructed by carefully monitoring the user in an unobtrusive way (assuming that the user agrees with such far-reaching measures). Modeling a person requires a rich and steady flow of information elicited in various ways on the one hand and a specifically tailored model or container in which the richness of the information stream can be captured on the other. Such a model would allow us to deduce what the person desires most at any time in any kind of situation with respect to time, geographical location, mood, environment, etc. While the scenario remains futuristic, when we look at services like last.fm (proposing music as it learns your taste), Amazon (recommending books that you might like), etc. we see that the digital revolution is starting to take more and more steps towards this unified personalized environment. While some people are afraid that in the future machines will determine our tastes and likings, we think that such services could make our lives a lot easier, when done in a moderate and governed way. In the end it will save us time to do the things we really care about.

In this chapter, we propose a user model which can contain all relevant user data in a context-sensitive way. This model is filled by parsing user feedback which can either be provided explicitly by user actions as well as implicitly by monitoring the user's behavior. Further, we list a number of measures to overcome the cold start problem occurring when new users, with an empty profile, enter the system. In other words, this chapter addresses research questions 5A (How can we model relevant user data to support context-sensitive adaptation?), 5B (How can we obtain this user data encompassing both explicit and implicit data?) and 5C (How can we support new users who suffer from an empty user model?).
6.1 Introduction

Independent of the domain, to enable personalization, good knowledge about the user is key. Hence, next to data integration, user modeling is regarded as the second important pillar in our approach presented in Chapter 4. The better the understanding of what the user would appreciate at any given time, the better any automated process can estimate and fulfill this need. However, this user knowledge needs to be centralized and modeled in such a way that any given service acknowledging the model, can easily query or extract consolidated user information. The administration of such a model involves three important processes: 1) observational extraction and retrieval of as much user information as possible; 2) parsing this information and filtering out relevant patterns and statements; 3) exposing the resulting collection of relevant user facts to applications enabling them to personalize their services.

Just like a domain model models the general domain of an application, a user model is generally regarded as a digital representation of the user, i.e. how a system sees and ‘understands’ the user. Originally, user modeling originated from the field of natural language dialog systems [147]. Over the years many different implementations and user modeling systems have been developed and reviewed as can be seen in [145]. The three most important user model characteristics according to Kobsa include:

- Generality and domain independence: user modeling systems are required to be usable in as many application and content domains as possible, and within these domains for as many user modeling tasks as possible.
- Expressiveness: systems are expected to be able to express as many types of assumptions about the user as possible at the same time.
- Inferential Capabilities: systems are expected to perform all sorts of reasoning that are traditionally distinguished in artificial intelligence and formal logic.

All three of these items can also be placed in the context of the television domain. For example, we foresee that, considering the fast evolving character of TV related services and features, a general and flexible model which can easily be applied in sibling domains will be advantageous. User model information from applications in different domains could for example be incorporated or relations between them laid without complications, to the benefit of all. In the TV domain it is expected that users will be able to express their opinion on various concepts like genres, programs, series, persons, etc. Further, every such an expression of interest will be accompanied by contextual information as well as the degree of interest. For these reasons, it is fundamental to have a model which can materialize these complex user expressions flawlessly. Lastly, in the TV domain the majority of information obtained will be elicited from the user’s behavior. All the programs he or she watches, zapping between channels, skipping advertisements, etc. should be caught by the system. However, this huge amount of information can be so overwhelming that it becomes difficult to make direct assumptions. Only through reasoning and the detection of underlying patterns, relevant information can be extracted.

In the richly described literature dealing with user modeling, often the term user profile is used interchangeably with the term user model. However, we and others believe there is a small discrepancy between these two terms. In [148], Koch describes a user profile as a simple user model, while in [97] Fröschl argues that the difference between user profiling and user modeling lies in the different level of sophistication. We regard the user profile as a simple storage of user information as obtained without any further enrichment, interpretation or reasoning. It contains the information in its basic form, how we retrieve it from the user. The user model on the other hand, is the container comprising the information extracted from the user profile after a process of enrichments, filtering and interpretation. Or as [148] states: “the user profile is used to retrieve the needed information to build up a model of the user”. Our definition of a user model now becomes: The user model is a dynamic representation of the user aggregated by observing and learning from the user behavior as well as explicit user knowledge. In the end, the user model needs to allow us to do the right thing at the right time in the right way [94].
6.1. INTRODUCTION

6.1.1 Explicit vs. Implicit

Usually, user models consist of explicit user data. A user states that she is 45 years old, female, capable of speaking three languages, fond of tennis and further expresses her likings on items like books, movies, CDs, television programs, etc. This is all very useful information in terms of content filtering and personalization. However, all this information is elicited from the user by means of her telling us these particular facts. In other words, every statement in the user profile was obtained because the user specifically informed us. However, such an approach is vulnerable to the subjectivity of the user. Every time a user utters a reaction like for example rating a book, movie, news item, etc., this rating can be influenced by numerous factors. For example, the user’s mood, fatigue, enthusiasm, happiness etc. can all influence his or her judgement. Moreover, if you for example ask a user to rate twenty items, and one hour later you ask the user again to rate the exact same set of items, but in a different order, several items on that list will end up with two different ratings. This subjectiveness can be caused by several reasons which pertain more to psychology theory than computer science. However in [114], Harter investigated user judgements in information retrieval systems from a psychological viewpoint. When a user initiates a search action within for example a document retrieval system and is confronted with the results, he makes determinations, called relevance judgments, about the output. These judgements are then influenced severely by various parameters in the direct environment. Specifically, in the television domain, the problem is even more pronounced because people tend to be less proactive or leaning back than a person actively (or leaning forward) browsing the Web. This passive behavior can then lead to the user providing less accurate feedback. E.g. when watching television after a hard day’s work, people want to be left alone. In contrast, when they for example want to search/rate books in the Amazon Web shop, it is their own choice and thus they expect that they will be required to give feedback, which they will gladly do as accurately as possible.

In iFanzy, explicit feedback mostly consists of user ratings, because they are a direct reflection of the user’s opinion. However, in general we can define three main problems with explicit feedback, from which two were already identified in [146]. Firstly, the relevance of information is always relative to the (changing) information need of a user. Judgements of individual items are typically assumed to be independent when in fact they are not. If the first two results in a search action already satisfied the user, then a third article on the same topic may simply be rated lower because the information need is already less dire. Secondly, computer users are known not to be very good raters. Even more so, in [146] we read that users are generally very reluctant to perform actions that are not directed towards their immediate goals if they do not receive immediate benefits, even when they would profit in the long run. Thirdly, there is a big discrepancy between how users utilize a rating scale and how it was expected to be used. In the ideal case a rating scale provides a fine granularity e.g. a scale from 1 (I do not like it) to 100 (I love it!), allowing a user to exactly convey how he feels about a specific item. However, in reality, the more choices the user has, the more he is driven to extremes and in particular the positive part of the scale. In [211], Schwab et al. show that it remains a challenge to obtain enough negative ratings to acquire useful evidence for pattern detection. This conclusion was again confirmed recently by the rating behavior on YouTube movies. In Figure 6.1 we see the dispersion of the ratings uttered on YouTube movies. YouTube allows registered users to rate movies by means of a five star scale. As the figure shows, there is a clear trend that users mainly rate with extreme values (in this case 1 star (not good) or 5 stars (great!)). On top of that, the figure also shows that there are much more items rated positively, than there are negative ones.

Obviously, for various reasons mentioned above, we cannot rely on explicit feedback alone. Personalized systems filtering information that only use explicit ratings require a large number of ratings to remain viable [181]. Implicit feedback on the other hand covers all the information we can elicit from the user without him or her realizing it. In a sense this is much more reliable information as the user exhibits more natural behavior (a user does not know that he is monitored).

1http://www.youtube.com
2Data was provided by YouTube and published on http://youtube-global.blogspot.com/2009/09/five-stars-dominate-ratings.html
However, technically it introduces extra complexity to monitor a user unobtrusively, let alone potential privacy issues. For the television domain on the other hand, implicit feedback is even more essential. Encouraging a television viewer too much to utter program ratings is difficult and potentially makes the user dislike the system. Implicit feedback however in this case signifies monitoring the user’s watching behavior alongside other actions like setting favorites, remainders, alerts, etc. Storing which programs were watched and for how long is not intrusive and, when anonymized properly, not considered a violation in the user’s privacy. Therefore, gathering implicit feedback can be considered as an essential part of building a rich user model. Moreover, in the television domain it would be indispensable.

6.1.2 Context

User behavior is a complex mechanism to unravel. Based on numerous parameters like mood, location, time, environment, people around, noise and light levels, health, etc., people can behave very differently and unpredictably at any given time. It is a utopia to believe we will ever be able to grasp these complex behavioral patterns completely. Still, even a simplified representation of this behavior can already help a great deal. E.g. if a user really loves to watch action movies, and always watches them during the weekend in the evenings, it would seem ridiculous to recommend him these movies at 8 o’clock in the morning when he would probably appreciated the weather and traffic reports more. The constrained setting in which one specific statement is valid is called the context. Only in the context “during evenings in the weekend” is the statement “user likes action movies” valid. Applications dealing with context are denoted by the term context-aware as firstly introduced in [208].

In broader sense, context can be seen as a description of the physical environment of the user on a certain fixed point in time. Some contextual aspects are particularly important for describing a user’s situation when dealing with television programs:

- **Time:** In which time frame was the explicit/implicit feedback valid? E.g. evening, morning, weekend, summer, workday, etc.

- **Location:** Where was the user residing when the explicit/implicit feedback was given? E.g. in the car, at home, at work, on vacation, etc.

- **Audience:** Which other users were part of the experience when the explicit/implicit feedback was uttered? E.g. with family, friends, just your partner, alone, etc.

- **Device:** Which devices was the user using at the time explicit/implicit feedback was uttered?

Note that context can be interpreted very broadly. Potentially we can for example also take the user’s mood, devices, lighting, noise level or even an extended social situation into consideration. Where this in theory indeed could potentially improve the prediction of what the user might appreciate to watch, in practice measuring all these states is considered not very effective with current technologies and sensor capabilities.
Working with context is always constrained by the ability to measure it. While exact location is hard to measure, from the personalization perspective it would be interesting to know the user’s exact location e.g. to give personalized information about that location. In case of the current audience, knowing who is watching is necessary to know how to deal with given user feedback. When for example a rating is given when several people are watching television, this might mean that this rating can be considered, to some extent, valid for all participants. Of course this requires that the audience can inform the system about their presence, either by explicit login via the remote control or implicitly e.g. via a RF-ID tag. Note that the audience aspect of context is only interesting in a multi-person environment, using for example a mobile phone is mainly considered an individual and personal activity.

Also more exotic forms of context, like mood and emotional state, have previously been researched. In [15] for example, Aroyo et al. present the move.me prototype which illustrates a scenario for social interaction in which users can manipulate audiovisual sources presented on various screens through an interaction with a sensor-enhanced pillow. Sensor feedback of the pillow was sent and maintained in the user model. One of the scenarios for example included changing the television program if it made the user too anxious or afraid (measured by skin conductivity sensors registering the user sweating).

Context is already regarded as an indispensable part of a personalized environment. In different domains like for example restaurant recommenders, traveling, networking cars, etc., people argue that not taking situational information into account, seriously limits the relevance of the results [261, 258, 237]. For more information about context and context-aware computing, we would like to refer to [77] and [81].

6.1.3 Semantic Web Technologies

Semantic Web is often considered as a supporting technology for adaptation, personalization and various Web services. Therefore, it is also often employed in user modeling or services required to model persons and their whereabouts in terms of friends, preferences, interests, activities and actions. Over the years, there has been some effort to model users or at least parts or aspects of users by means of Semantic Web technologies.

**GUMO** (General User Model Ontology) is a general user model ontology for the uniform interpretation of distributed user models in intelligent Semantic Web enriched environments [118]. GUMO is modeled in OWL and influenced by older technologies like the XML based **UserML** (User Modeling Markup Language), **SUMO** (Suggested Upper Merged Ontology), etc. GUMO contains numerous classes to model the user’s characteristics, preferences, emotional state, personality, etc.

**FOAF** (Friend Of A Friend) is an ontology describing people, the relation between people, the relation between people and objects like publications, Web pages, mail boxes, etc., activities, interests, etc. [49]. FOAF is modeled in OWL and enables people to publish descriptive information about themselves onto the Web, in a machine-readable way. Because of the ability to create relations between people, networks or communities can be formed without requiring a central authority or server.

**SIOC** (Semantically-Interlinked Online Communities Project) is an RDFS-based ontology, which combines terms from vocabularies that already exist with new terms needed to describe the relationships between concepts in the realm of online community sites [47]. The goal of SIOC is to interconnect different discussion primitive like bulletin boards, weblogs and mailing lists from different online communities. SIOC will facilitate the location of related and relevant information; by searching on one forum, the ontology and interface will allow users to find information on forums from other sites that use a SIOC-based system architecture. Other uses include cross-site querying, topic-related searches, and the importing of SIOC data into other systems. SIOC can be integrated easily with FOAF by means of the **FOAF:holdsAccount** property.

In the following section (6.2) we introduce our events-based contextualized user model. Afterwards in Section 6.3 we explain the cold start problem together with some measures to counteract its effect. Conclusions come at the end in Section 6.4.
6.2 The User Model

In this section we take a deeper look into the user model structure. In Subsection 6.2.1 we look at the general structure of the model and illustrate it by means of an example. In Section 6.2.2 we show that the generation of user events serves mainly as input to the user profile. Every event is contextualized by means of an auxiliary context model. Finally in Section 6.2.4 we elucidate further how we take the events from the profile and create a consolidated user model by means of data mining techniques.

6.2.1 The User Model Structure

In Figure 6.2 we see the RDFS/OWL schema of the user model. Just like with the content model, we divided this schema in different layers: the top layer indicates external classes from which iFanzy classes inherit semantics, the bottom layer shows iFanzy classes. To assure a wide and flexible applicability, we kept the model simple but powerful, able to express almost any kind of user information. Centrally in the iFanzy layer we see the class IFZ:User which inherits semantics from the FOAF person class FOAF:Person and the SIOC user class SIOC:User. This inheritance effectuates that every existing FOAF of SIOC user description automatically becomes an instance of the iFanzy user class, facilitating a smooth integration of such external person descriptions. Further, the user model also provides a class IFZ:Login to maintain login information of different sources where the user might have an account. To keep track of characteristics (like e.g. height, weight, age, address, gender, etc.) on the one hand and preferences (like e.g. preferred device, music loudness, screen brightness, type of pizza, etc.) on the other we introduced the classes IFZ:Characteristic and IFZ:Preference which are both subclasses of IFZ:Property. This property class has the ability to very generically model any piece of information which would apply as a property to the user. The class IFZ:Property has a property name, data type and value. In Figure 6.3 we see three instances of this property class. The first instance (at the top of the figure) models the user preferred mobile device. The value of this property is DEV:Nokia6021 where ‘DEV’ is here regarded as the namespace of an example ontology of devices. The data type is a more general class modeling various mobile devices. The second instance (in the middle of the
6.2. THE USER MODEL

Figure 6.3: Instance Example of the Property Class

The figure models the age characteristic of a user who is 28 years old. The data type of the property is in this case XML:Integer. Lastly, the third instance (at the bottom) models the user’s place of birth, which is ‘Antwerp’, modeled by the IFZ:Location data type. Generally, the data type of the property here allows for the interpretation of the value. Moreover, any single class from any single ontology can be used as data type. In principle, this structure allows for every possible characteristic or preference to be materialized in the user model.

Besides characteristics and preferences, the user model also needs to provide a structure to express valuable user feedback related to any type of concept. A user can state for example that he likes Cuba (a location), The Godfather (a movie), Friends (a series), Al Pacino (an actor), etc., together with an indication how much he or she likes that concept. But also broader, a user can rank a set of options, declare a concept as favorite or send a references to a friend as a tip. Basically, through different user interfaces a user can perform a whole set of actions which are relevant to be maintained. In our schema as shown in Figure 6.2, we model actions as assertions (by means of the IFZ:Assertion class), to say for example: “The user asserted that he rated “The Godfather” with a ‘9’”. Every assertion also applies on a specific resource indicated by the property IFZ:onResource. In Figure 6.4 we see some examples of types of assertions and resources currently used in iFanzy. At the top, we see the assertion and resource classes as also shown in Figure 6.2. On the left, we see subclasses of IFZ:Assertion, like IFZ:Rating (all rating assertions), IFZ:Reminder (assertions that a reminder was set), IFZ:Favorite (all favorites set), etc. On the right, we see the relevant resources (subclasses of RDFS:Resource) like a program, a program group (in TV-Anytime the concept to model a series), persons, genres and service information types (in TV-Anytime, a service is whatever is broadcasting the content, e.g. a television channel). All of these can be used by the assertions and if ever necessary, thanks to the properties of the RDF language, more can be added since everything is a subclass of RDFS:Resource. The relation between an assertion and a resource is IFZ:onResource. However, for every type of resource we can use a different property name better indicating the semantics of the property. For example, the property between a rating assertion and a program resource is called IFZ:onProgram. All these specific properties are then subproperties of IFZ:onResource as indicated by the RDFS:subPropertyOf relation. Note that the class IFZ:Person differs from IFZ:User. Persons also includes actors, directors, presenters, etc. while the user class models an iFanzy user.
Every assertion models user feedback on a specific resource defined in the data model. However, because this data model connects different resources by means of semantic relations, we can exploit the graph by reasoning over them. If a user for example indicates that he likes program \( P \) and \( P \) is richly annotated in the data set, we can exploit the relations between \( P \) and any other resource. If through the structure of our model we know that \( P \) is related to program \( Q \) and that person \( X \) represents the central character in \( P \), the feedback of the user can therefore also influence \( Q \) and \( X \). More practically, if the user tells us that he likes The Netherlands (which is modeled by a geographical location concept), then we can reason that he would probably also like Amsterdam because its concept has a \textit{partOf} relation to the Netherlands. Therefore, by exploiting the graph, one user assertion can lead to the deduction of many more interest predictions.

\subsection*{6.2.2 Context-sensitive User Generated Events}

As explained in the previous subsection, in iFanzy every user generated statement is regarded as an \textit{event} (see Definition 6.2.1). Pressing a button on the remote control, a search action on the Web portal, rating a program, watching a program, etc. is all considered a user event. Some events are triggered implicitly, others explicitly. Essentially, the user performs some kind of action which generates an event, from which in turn we want to learn as much as possible. However, until now the model did not consider any context. Yet, context is of utmost importance to understand what, when and how a user likes different styles of programs. Therefore, every generated user event is accompanied by a context describing the situation in which the event occurred.

\begin{definition}
An event \( E(u, r, c) \) represents an action generated by user \( u \), involving resource \( r \) under context \( c \).
\end{definition}

All generated events together encompass all the information we obtained from the user, which therefore shapes the user’s profile. In Figure 6.5 we see the RDF(S) schema of an event including an associated context. Each event (modeled by the IFZ:Event class) has a type IFZ:eventType (e.g. ‘WatchEvent’, ‘RateEvent’, ‘AddToFavoritesEvent’, ‘RemoveFromFavoritesEvent’, etc.), one or more properties, and occurs in a specific context. The IFZ:EventProperty is again a subclass of IFZ:Property which was already shown and explained in Figure 6.2 and 6.3. The IFZ:Context class has four properties which model contextual aspects as explained above. The IFZ:onPlatform property contains the platform from which the event was sent, IFZ:onPhysicalLocation refers to a geographical concept which will only be filled in once we are able to accurately pinpoint the user’s location. The IFZ:hasTime property tells us the time of the event by referring to the OWL Time ontology and with the IFZ:hasParticipant property we can maintain all the users which were involved in this event, representing the audience.

All the information we aggregate from the various events, is materialized in the user model as shown in Figure 6.6. In this sense, we can see all generated events (the user profile) as the short-term ‘memory’ where every input is saved, while the user model signifies the long-term ‘memory’
6.2. THE USER MODEL

where only useful data is saved, filtered from the user profile. After a set amount of time, old events are removed.

In the user model we have a list of assertions (IFZ:Assertion) which are extracted from the events generated by the user and act on a certain resource (RDFS:Resource). Above the line we see the user profile, listing the events which were fired together with the context they occurred in. Below the line we see the influence of this event on the user model. From the event that occurred, a new assertion is created to represent the effect on the specific resource. For example a rating event on a program \( P \) causes the creation of a rating assertion on resource \( P \).

However, not every event is translated into an assertion in the user model. Sometimes an event is not relevant enough (e.g. a ‘WatchEvent’ where the user watched a program for ten seconds and then zapped away). On the other hand it might also be possible that multiple events are aggregated into one user model update, like when detecting a certain pattern of events that might be worth drawing conclusions from (e.g. a ‘WatchEvent’ on the same program every week). However, one event can also potentially lead to the creation of various assertions on different resources, the property IFZ:forResource therefore indicates the resource on which the original event was fired.
The aggregation of assertions in the user model can be seen as a filter over the events, and the collection of events as the history of all user actions. For this aggregation we have several different strategies depending on the event type. A rating event on a specific resource for example always creates a new assertion instance overwriting any previous rating assertion on that same resource.

In iFanzy we currently distinguish eight different assertions, which can be seen in Figure 6.7. Seven of these can be triggered by a user generated event. E.g. an ‘addToFavorites’ event can lead to a ‘Favorites’ assertion in the user model. Assertions which can be triggered by the user himself include:

- **Selection**: The user can select a program to be added to his or her daily TV program selection. In this way, a user can compose the programs he or she wants to see that particular day.
- **Reminder**: The user can set a reminder on a program or a series, to be notified when that program or an episode of that series is about to start.
- **Tip**: The user can send program tips to their friends.
- **Recording**: The user can schedule a program or series to be recorded.
- **Ranking**: The user can rank TV channels, indicating a preferred priority.
- **Rating**: Objects like programs, channels, genres, etc. can be rated.
- **Favorite**: The user can add different concepts to his or her favorites. Favorites can be edited by the user and represent the program subset he or she really appreciates.

In Figure 6.7, we can see to which types of resources these assertions can be applied. The rating assertion can for example be applied to different kinds of resources including programs, channels, series, persons and genres. A selection assertion on the other hand can only be applied to a program resource. All the properties shown in Figure 6.7 are subproperties of IFZ:onResource.

To illustrate, in Figure 6.8 we see a small part of a user model (do note that we do not show any contextual information in this graph, which would normally be included in the user model). For the sake of the example, imagine that the user fired implicitly a ‘rankChannelEvent’ on the BBC One channel, a ‘addReminderEvent’ on the second season of ‘Blackadder’ and an explicit ‘rateEvent’ on the third episode of that season. These three events led to the user model shown in Figure 6.8, where three according assertions were added. The ranking assertion IFZ:ranking1 shows that the BBC One channel was moved to the first place in the EPG (which in turn also automatically updated previously asserted rankings of channels which now drop one place). The IFZ:rating23 assertion shows that Blackadder episode number three of season two was rated with a score of eight in the interval [0, 10]. Lastly, the IFZ:reminder2 assertion shows that the user wants to be reminded every Monday at 8pm if some episode of the second season is programmed.
6.2. User Model

6.2.3 User likings

So far, we saw that users can set a rating on programs and that implicit events are generated for all different kinds of actions, including for example setting favorites, reminders, etc. However, all these implicit assertions also at least indicate some kind of interest from the user. If the user ranks his channels, putting BBC on top, indicates a potentially positive interest in BBC programs. The fact that the user for example wants to be reminded about the start of a program or wants to records one, also indicates a potential interest. This ‘interest’ is materialized via the one assertion which can not be directly triggered by the user, namely the liking assertion (see Definition 6.2.2).

This assertion is used by the system to indicate that it could deduce that the user probably likes a particular concept or resource.

Definition 6.2.2. Liking

A liking $L(u, r, c, s)$ represents the degree of potential interest of a user $u$ in resource $r$ under context $c$, as determined by source $s$.

In Figure 6.9 we see three liking assertions which were deduced from the previous three assertions shown in Figure 6.8. Ranking a channel to the first place indicates that the user must have some kind of interest in this channel, and therefore led to the generation of a liking of ‘7’ in this channel. Note that a liking class has a source property (IFZ:source) indicating the type of the source which led to the generation of this liking (in this case the source was a ‘ranking’ event). This additional property was necessary because events are removed after a specific time (since they represent only the short-term memory as previously explained), and this source information also remains relevant in the long-term memory. Although not shown in Figure 6.9, every liking also maintains the context in which it was valid.

In Figure 6.9, we see that rating the Blackadder episode led to a liking on the genre ‘Comedy’ (IFZ:3.5.7), with a value of ‘8’ (the same as the rating) and a source ‘rating’. Lastly, setting a reminder on the second season of Blackadder effectuated a liking on the series, with a value of ‘7’ and a source ‘reminder’, as well as another liking on the genre ‘Comedy’ with the same source and value. The initial value chosen for a newly created liking, represented by $L_0$ and currently set to ‘7’, was chosen based on user study research, and will be further elucidated in Chapter 10.
CHAPTER 6. USER MODELING

Figure 6.9: Newly deduced liking assertions

Whenever a second or more likings are generated from the same source on the same resource, the value is updated. Hence, a liking represents an evolving prediction of the system representing how a user might feel about that concept. This update increases the value with \( L_i \), which is an amount depending on the type of the event (or source) which caused the liking. After all, some events are better in predicting the user’s interest. Similarly, the liking’s value can also be decremented with \( L_i \) when a user for example removes a program from his favorites. Further, likings which haven’t been updated for a long time, are eventually removed from the user model.

In general, every event \( E(u,r,c) \) causes the generation and/or update(s) of a set of likings following Formula 6.2.1.

**Formula 6.2.1. Generation of likings from an event**

Let \( r \) be a resource, \( A = a_1, \ldots, a_n \) be the set of individuals which are connected to \( r \) by object properties, \( E(u,r,c) \) an event, \( t \) the type of this event, \( L_0 \) and \( L_i \) the start and increment values of a liking generated by event \( E \). Then:

\[
L(u,r,c,t) = \begin{cases} 
L_0 & \text{if } L(u,r,c,t) \text{ did not exist} \\
L(u,r,c,t) + L_i & \text{otherwise}
\end{cases}
\]

\[
\forall a \in A : L(u,a,c,t) = \begin{cases} 
L_0 & \text{if } L(u,a,c,t) \text{ did not exist} \\
L(u,a,c,t) + L_i & \text{otherwise}
\end{cases}
\]

The first part in Formula 6.2.1 shows that the resource on which the event was triggered, receives a liking \( L_0 \) if it did not exist yet and otherwise an increment of \( L_i \) if it did exist. Further, the second part of Formula 6.2.1 shows that all individuals connected to the resource, via an object
6.2. THE USER MODEL

property, also receive this liking/increment. For example, as shown in Figure 6.9, the reminder
which was set on the series Blackadder also led to the creation of a liking of that series’ genre. In
this way we exploit the semantic data graph to propagate likings to related resources.

However, some ‘special’ events divert from this general formula. Firstly, the ‘sendTip’ event is
special in the sense that sending a program as a tip to a friend does not say anything about your
interest in that program. After all, it is possible that you know your friend will like a particular
program, while you yourself do not. Secondly, the ‘rate’ event is special because the user himself
provides a value, whereas in the general approach every event results in positive likings. Therefore,
for every ‘rate’ event a different approach applies like shown in Formula 6.2.2.

Formula 6.2.2. Generation of likings from an ‘rate’ event
Let \( r \) be a resource, \( A = a_1, \ldots, a_n \) be the set of individuals which are connected to \( r \) by object
properties, \( E(u, r, c) \) a ‘rate’ event of user \( u \) in context \( c \), \( t \) the type of this event (here ‘rating’) and \( v_r \) the value of the rating on resource \( r \). Then:

\[
\text{IF } L(u, r, c) \text{ exists THEN remove } L(u, r, c)
\]

\[
\forall a \in A : \begin{cases}
    v_r & \text{IF } L(u, a, c, t) \text{ did not exist} \\
    \text{AVG}(L(u, a, c, t), v_r) & \text{Otherwise}
\end{cases}
\]

Formula 6.2.2 clearly differs from the general approach. Firstly, if the user rates a resource,
every possibly liking in the same resource (and context) is removed, independent of the source.
After all, we now exactly know how the user thinks about this resource, making the liking su-
perfluous. The second part again looks at all related resources connected via an object property.
However in this case, we calculate the average between the new rating value and the old liking of
that resource. Hence, if the user consequently rates all football programs with a value ‘0’, because
he completely dislikes it, then the liking in genre ‘Football’ should also evolve towards ‘0’.

Since all resources in the data graph are interrelated, propagation of these likings can deliver
some interesting unexpected results increasing the chance of serendipitous recommendations in
the future.

6.2.4 Mining User Event Patterns

The user profile, which consists of all the events either explicitly or implicitly triggered by the
user, can easily grow to huge proportions. Especially if we could, for example through a set-top
box (STB), monitor the watching behavior and thus would obtain events every time a user zaps
around. In effect, only a subset of these events is found truly valuable and worthy of materializing
in the user model as explained in the previous subsections. However, sometimes hidden patterns
occur within the event set which were not found when going over the set event by event. What
we are mainly looking for in the user profile are specific recurring patterns. These patterns are for
example interesting to discover likings about concepts which the user did not mention explicitly.
If we see in the user profile for example, that the user watches ‘Friends’ every Monday evening
without explicitly stating that he or she likes that program, the system can make that assumption
itself by adding an appropriate ‘liking’ assertion.

Data mining is the general term for the process involved in detecting patterns in large quantities
of data. With the recent unbridled growth of databases in all kinds of domains, data mining
manifests itself more and more as the best approach for discovering and elucidating the patterns
that underlie it [257]. A field closely related to data mining is Machine learning. Machine learning
is the field which deals with contriving algorithms which allow a computer to ‘learn’ particular
patterns based on a set of inputs. Although the term ‘learning’ might seem a bit strange for a
computer (it is not really getting smarter), in general ‘learning’ here is interpreted as the ability
to change the behavior of the system making it perform better in the future. Over the years, an
enormous number of machine learning algorithms have been developed. The data to be investigated
by an algorithm consists of a set of instances (or items) which are described by a number of attributes (or features). In general, learning algorithms can be classified in four different styles:

- **Classification learning**: Algorithms which try to find the outcome of a target class, provided a set of training examples. A well-known example is the weather problem, where the weather is described by four parameters (outlook, temperature, humidity and wind) and a target class which shows whether or not the weather allows to play an outdoor game. The algorithm learns a model using the training set which shows a number of weather situations in which it was already decided to play or not to play. After training new weather situations are presented, but now the trained algorithm should predict whether or not the weather allows playing the game. Example algorithms include *NaiveBayes*, decision tree algorithms like *J48*, rule-based classifiers like *JRip*, etc.

- **Association learning**: Association learning is similar to classification learning, with that difference that there is no target class specified. Association learning algorithms try to discover any structure in the data that might be ‘interesting’ by finding a target class itself. Therefore, every attribute or combination of attributes in the data can now become a target class. An association rule for the weather problem (described above) could for example include: If temperature is ‘cool’ then humidity is ‘normal’. A well-known association learning algorithm is the *Apriori* algorithm.

- **Clustering**: Clustering algorithms try to group instances of the data, that seem to fall naturally together, based on the values of the features. Imagine a long list of elephant instances with features like height, weight, coloring, skin type, ear type, etc. A clustering algorithm is then able to construct different clusters or groups which represent the different kinds of elephants currently existing (e.g. African elephant, Indian elephant, etc.), based on these features. Examples of clustering algorithms include the *K-Means*, *DBScan* and *EM* algorithms.

- **Numerical regression**: Numeric regression algorithms are a variant of classification learning in which the outcome is a numeric value instead of a category. If we for example slightly change the weather problem, not to predict whether or not to play but rather the time (in minutes) to play the game, it becomes a numeric prediction. Examples include the *Linear Regression* algorithm.

Learning problems are also sometimes said to be *supervised* or *unsupervised*. Mostly, classification learning and numerical regression are called supervised because the algorithm learns from already correctly classified examples. Hence, the operator provides classified instances and thus ‘supervises’ the process. Clustering algorithms and association learning on the other hand, are called unsupervised because they learn patterns without any given input.

To discover patterns from the event data in the user profile, we focus on classification learning and association learning. Classification learning can help in predicting how a user might feel about a program that is about to be broadcast (a new example) based on previous ratings and likings on older programs (the classified examples). In this case, the rating is the target class. Association learning can help us by discovering that if a program has the genre ‘football’ and the user rated it “very high”, then the broadcast channel was ‘RTL4’. This association shows us that the user prefers to watch football on this particular channel. For this specific rule, the channel was picked as target class.

From the user profile, the most useful events with respect to data mining include events displaying the user’s viewing behavior and uttered ratings. These are most usable because firstly they give a clue on the user’s program interest, and secondly they occur most often in the list of user generated events. Other events indicating a user’s interest like for example setting favorites occurs considerable less often yielding a too small set of statements. In the future, realtime viewing behavior (which would be captured for example on a set-top box) would also become an interesting source of events. Currently, in iFanzy such sets are not available yet. Therefore, for the moment we only consider user rating events.
6.2. **THE USER MODEL**

<table>
<thead>
<tr>
<th>Channel</th>
<th>Format</th>
<th>Genre</th>
<th>Day</th>
<th>Frame</th>
<th>Series</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ned1</td>
<td>2.1.1</td>
<td>3.1.1.1</td>
<td>Fri</td>
<td>Evening</td>
<td>NOS-Journaal</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>(Bulletin)</td>
<td>(News)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NG</td>
<td>2.1.4</td>
<td>3.1.6.2</td>
<td>Wed</td>
<td>Noon</td>
<td>Earth Investigated</td>
<td>Very high</td>
</tr>
<tr>
<td></td>
<td>(Docu)</td>
<td>(Nature)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BBC2</td>
<td>2.4.2</td>
<td>3.5.2.1</td>
<td>Mon</td>
<td>Evening</td>
<td>Eggheads</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>(Show)</td>
<td>(Quiz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: The most relevant features of a rating event

iFanzy currently allows people to rate a program with a value in the interval \([0, 10]\) where 0 denotes “I hate this” and 10 indicates “I love it!”. The most relevant program features we can further include are (next to the rating itself): channel, format genre, content genre, day of broadcast, day frame and the series it belongs to. There are also other features imaginable (like e.g. actors, directors, presenters, image quality, etc.), however due to the fact that they occur much less frequently in the metadata, these were not included as they would make the data very sparse. In Table 6.1 we see an example of these features listed for three rating events. The table can only contain one format and one content genre. These genres are terms from the genre classification hierarchies which consist of a great number of different genres (800+). However, it can occur that a program has more than one genre associated for either or both the format and the content attribute. In such a case, we choose the genre which occurs most frequently in the whole program set. If we need to choose between genres which occur comparably often in the program set, we look at their depth (genre 3.4.6.7 (science fiction) has a larger tree depth and thus is more specific in the tree than genre 3.4 which represents “Fiction/Drama”) in the genre tree, and choose the one which is most specific. Note that we could also have chosen to take every possible genre (around 800+ in TV-Anytime) as a binary feature of the program. However since programs on average only have about 3 associated genres, this could lead to too many degrees of freedom and thus overfitting of the model. Further, we also show the time of broadcasting of the program, albeit represented as a day frame like e.g. afternoon, evening, etc. The reason for this is twofold. Firstly, data mining algorithms perform better with features which are limited in their possible space. In our case we have split a day in seven well-chosen frames (dawn, morning, noon, afternoon, evening, midnight and night), which is for example much less than the 1440 options we would have if we round the start time of a program to the minute. Secondly, due to the regular time schema most people live their lives in, these frames are chosen in such a way that they represent the different behaviors exhibited by a person during the day. Similarly, the program rating (in \([0,10]\)) is also converted into a class scale including classes “very low”, ‘low’, ‘medium’, ‘high’ and “very high”.

To perform our experiments, we make use of the *Weka* data mining suite [257]. The Weka workbench is a software suite facilitating a huge number of data mining algorithms. Weka can preprocess data, evaluate learning schemes and visualize both data and learned patterns. Because of the variety and number of different data mining algorithms, it is an extensive task to select those which match best with the current problem description and data structure. In our particular case we experimentally selected three different algorithms which proved to be good learners for the user’s rating behavior data set. These algorithms in turn could then help us in predicting the user’s future program interests. To load the data into the Weka suite, the *Attribute-Relation File Format* (ARFF) was used. In Figure 6.10 we see the ARFF entries for the three examples shown in Table 6.1.

To train the classification algorithms, a training set (containing about 50% of the statements) was randomly selected to train the algorithm before it was executed on the remaining part. In this way we can validate that the rules found in the training set also apply on the remaining part for the data. We tested the algorithms on the ratings of ten iFanzy users we picked due to their relatively high number of uttered ratings (at least 200). Do note that we suffer here from a subtle sample bias. These users did not rate just a random selection of programs, but rather
programs which they either liked or where they did not agree with the shown (predicted) rating. Still, because it reflects the user’s preferences, this data allows us to determine his or her program taste.

To validate the outcome of the data mining algorithms, we compare their results to the baseline prediction. For this test, the baseline is set exactly as the score of the classifier that always predicts the largest class. In our case, the classifier which always predicts “very high” for every program. When running this classifier for the selected users, on average 30% of instances was classified correctly. Therefore in every subsequent test, this will be the minimum every classifier should be able to reach. The three following algorithms performed best on the validation set after training on the training set.

JRip

JRip is a rule generating classification algorithm. This algorithm tries to find a rule which covers as many instances of a class as possible while excluding instances which are not in that class. After the creation of such a rule, the covered instances are removed, and the process starts again with the remaining instances. In this way a set of rules is obtained which together cover the largest possible number of instances. JRip implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), including a heuristic for global optimization of the rule set [65].

To test our algorithms, we selected ten iFanzy users which we consider to be the most reliable. These ten are users which use iFanzy on a regular basis and previously indicated in user tests that they like the iFanzy concept. Because of their regular usage they also have the largest set of generated rating events which is necessary for every data mining algorithm trying to find well-supported patterns. When we execute the JRip algorithm on the rating events of these ten best iFanzy raters, we get on average 59.4% correctly classified instances (and thus 40.6% incorrectly classified) which is slightly less than twice the baseline’s performance. Further we see a Kappa statistic (a measure of inter-rater agreement, ranging from 0 (no agreement) to 1 (perfect agreement), while correcting for agreement that occurs by chance) of 0.3386 (Fair agreement). The Kappa statistic thus indicates that the predicted value and the correct value here agree fairly good. The Mean Absolute Error metric, a quantity used to measure how close predictions are to the eventual outcomes, returned 0.2218 where of course 0 is the target value. Among the rules found by JRip for one out of the selection of ten users, we see for example:

@RELATION rating_events_user_10

@ATTRIBUTE channel {Ned1, Ned2, NG, Disc, BBC1, BBC2,...}
@ATTRIBUTE format {2.1, 2.1.1, 2.1.4, 2.4.1, 2.4.2,...}
@ATTRIBUTE genre {3.1, 3.1.1, 3.1.1.1, 3.1.6.2, 3.5.2.1,...}
@ATTRIBUTE day {monday, tuesday, wednesday, thursday,...}
@ATTRIBUTE dayframe {morning, noon, afternoon, evening,...}
@ATTRIBUTE title {1, 2, 3, 4, 5, 6,...}
@ATTRIBUTE rating? {'very low', low, medium, high, 'very high'}

@DATA
Ned1, 2.1.1, 3.1.1.1, friday, evening, 3, 'high'
NG, 2.1.4, 3.1.6.2, wednesday, noon, 105, 'very high'
BBC2, 2.4.2, 3.5.2.1, monday, evening, 581, 'low'
...
6.2. THE USER MODEL

(day = friday) and (dayframe = evening) and (channel = Ned2) => rating?=high
(day = saturday) and (dayframe = evening) and (format = 2.6) and (content = 3.4.6.4) => rating?=‘very high"
(channel = RTL4) and (day = tuesday) and (dayframe = evening) => rating?=‘very high"

These rules now tell us that based on previous ratings, this user enjoys his Friday evening watching programs broadcast on Ned2, likes detective movies on Saturday evening and RTL4 programs on Tuesday evening. These rules can now be translated into assertions which can be added to the user model. We can for example add the assertion that the user has a reasonable liking in the channel ‘Ned2’ in the context of Friday evening. Or that the user had a considerable liking in the channel ‘RTL4’ in the context of Tuesday evening. If in this context another liking already existed from the same source (here JRip), the new one overwrites it. Because our assertion model works with numerical values in [0, 10], we have to transform the classes ‘low’, ‘medium’, ‘very high’, etc. back to a numeric liking value.

NaiveBayes

The classifier called naive Bayes is a simple probabilistic classifier based on applying the Bayes’ theorem. This theorem postulates that the probability of the occurrence of an event is related to the occurrence or non-occurrence of an associated event. The naive Bayes classifier is termed naive because it relies on two important simplifying assumptions: 1) it assumes that the predictive attributes are conditionally independent given the class, and 2) it posits that no hidden or latent attributes influence the prediction process [140]. Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised classification learning setting, which we have.

On execution of this algorithm on the rating events of the ten best iFanzy raters, we get on average 60.4% correctly classified instances (and thus 39.6% incorrectly classified) which is over twice the baseline’s performance. Further we see a Kappa statistic of 0.3871 (Fair agreement) and 0.1914 for the Mean Absolute Error (MAE) metric. However, the most useful output information of the naive Bayes algorithm, is the table showing the strengths of the relations between the distinct attributes and each of the five target classes (“very low”, ‘low’, ‘medium’, ‘high’, “very high”). Because the naive Bayes algorithm considers every predictive attribute independently, it also lists these strengths per attribute. In Figure 6.11 we can see these strengths between on one hand the target class (on the horizontal axis), and on the other every possible value in every one of the six features (on the vertical axis), for one particular user.

If we look for example at the crossing of “very low” and the channel “Eurosport 2” in this matrix we see the value ‘24’, which shows us the number of programs which were broadcast on “Eurosport 2” and rated “very low” by the user. This table can now be used to predict how a new program would be liked by the user by means of the naive Bayes probabilistic model. To calculate for example the probability that a new program P, characterized by the features $(F_1, \ldots, F_n)$, will be classified in the class “very high”, we make use of Equation 6.1. In Equation 6.1, $Z$ is a scaling factor which depends on only $(F_1, \ldots, F_n)$, i.e., a constant if the values of the feature variables are known. In our case all values are always known (in case of a new title, we add it immediately), and therefore we choose $Z$ equal to 1, eliminating the fraction.

$$p(C|F_1, \ldots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C) \quad (6.1)$$

Imagine now a new program (not rated before) which is characterized by the following feature vector (‘RTL4’, ‘3.2’, ‘2.1.5’, ‘Tuesday’, ‘evening’, ‘X’) which translates to a talk show about sports named ‘X’, broadcast on RTL4 on a Tuesday evening. If we apply Equation 6.1 we can calculate the probabilities for the five possible target classes. Figure 6.12 shows this calculation for the class “very low”. In Table 6.2 we see the results for all five classes, from which we can now conclude that
class “very low” has the highest probability and thus program \( P \) will probably not be liked very much by this user. Looking back at Figure 6.11, we see that the user consequently rated sports (3.2) programs with rating “very low”, which in turn was also the decisive factor for this new prediction. The equation which can be used to classify \( X \) is shown by Equation 6.2. It calculates the prediction for every class (in our case five), and classifies the program in the class with the highest prediction.

\[
\text{classify}(f_1, \ldots, f_n) = \arg\max_c p(C = c) \prod_{i=1}^n p(F_i = f_i | C = c) \tag{6.2}
\]

Concluding, we can say that the naive Bayes algorithm is able to help a great deal in predicting a possible classification for new programs based on previously uttered user ratings. Further, given that this algorithm scores an average of 60.4% correctly classified instances, we can say that this is a good learner for this particular prediction problem.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>very low</th>
<th>low</th>
<th>medium</th>
<th>high</th>
<th>very high</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability</td>
<td>0.4</td>
<td>0.1</td>
<td>0.05</td>
<td>0.1</td>
<td>0.34</td>
</tr>
<tr>
<td>channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTL4</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
<td>16.0</td>
</tr>
<tr>
<td>Eurosport 2</td>
<td>24.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Ned1</td>
<td>2.0</td>
<td>1.0</td>
<td>3.0</td>
<td>2.0</td>
<td>13.0</td>
</tr>
<tr>
<td>[ \text{TOTAL} ]</td>
<td>111.0</td>
<td>48.0</td>
<td>36.0</td>
<td>48.0</td>
<td>99.0</td>
</tr>
<tr>
<td>content</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4.6.4</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
<td>3.0</td>
<td>9.0</td>
</tr>
<tr>
<td>3.2</td>
<td>37.0</td>
<td>3.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>[ \text{TOTAL} ]</td>
<td>79.0</td>
<td>31.0</td>
<td>27.0</td>
<td>26.0</td>
<td>43.0</td>
</tr>
<tr>
<td>format</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.5</td>
<td>9.0</td>
<td>3.0</td>
<td>1.0</td>
<td>6.0</td>
<td>2.0</td>
</tr>
<tr>
<td>[ \text{TOTAL} ]</td>
<td>43.0</td>
<td>21.0</td>
<td>16.0</td>
<td>26.0</td>
<td>45.0</td>
</tr>
<tr>
<td>day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>monday</td>
<td>16.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>6.0</td>
</tr>
<tr>
<td>tuesday</td>
<td>10.0</td>
<td>6.0</td>
<td>4.0</td>
<td>5.0</td>
<td>19.0</td>
</tr>
<tr>
<td>[ \text{TOTAL} ]</td>
<td>91.0</td>
<td>28.0</td>
<td>16.0</td>
<td>28.0</td>
<td>79.0</td>
</tr>
<tr>
<td>dayframe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>noon</td>
<td>13.0</td>
<td>2.0</td>
<td>2.0</td>
<td>1.0</td>
<td>4.0</td>
</tr>
<tr>
<td>evening</td>
<td>68.0</td>
<td>19.0</td>
<td>9.0</td>
<td>22.0</td>
<td>70.0</td>
</tr>
<tr>
<td>[ \text{TOTAL} ]</td>
<td>88.0</td>
<td>25.0</td>
<td>13.0</td>
<td>25.0</td>
<td>76.0</td>
</tr>
<tr>
<td>title</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title12</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>6.0</td>
<td>2.0</td>
</tr>
<tr>
<td>title27</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>[ \text{TOTAL} ]</td>
<td>252.0</td>
<td>187.0</td>
<td>175.0</td>
<td>188.0</td>
<td>239.0</td>
</tr>
</tbody>
</table>

Figure 6.11: Naive Bayes network overview for user \( U \)
6.2. THE USER MODEL

\[
p(\text{"very low"} | \text{"RTL4"}, \text{"3.2"}, \text{"2.1.5"}, \text{"tuesday"}, \text{"evening"}, \text{"X"}) \\
= p(\text{"very low"}) \\
\times p(\text{"RTL4"} | \text{"very low"}) \\
\times p(\text{"3.2"} | \text{"very low"}) \\
\times p(\text{"2.1.5"} | \text{"very low"}) \\
\times p(\text{"tuesday"} | \text{"very low"}) \\
\times p(\text{"evening"} | \text{"very low"}) \\
\times p(\text{"X"} | \text{"very low"}) \\
= 0.4 \times (3/111) \times (37/79) \times (9/43) \times (10/91) \times (68/88) \times (1/253) \\
= 0.000000356
\]

Figure 6.12: Naive Bayes probability calculation example

<table>
<thead>
<tr>
<th>Class</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very low</td>
<td>3.56\text{e}^{-7}</td>
</tr>
<tr>
<td>Low</td>
<td>4.99\text{e}^{-8}</td>
</tr>
<tr>
<td>Medium</td>
<td>3.16\text{e}^{-9}</td>
</tr>
<tr>
<td>High</td>
<td>1.54\text{e}^{-9}</td>
</tr>
<tr>
<td>Very high</td>
<td>5.24\text{e}^{-8}</td>
</tr>
</tbody>
</table>

Table 6.2: Probability calculation result per class

Apriori

Apriori is an association learning algorithm. Its primary task is to learn association rules between different attributes of the data set. Usually this algorithm is applied in databases with huge numbers of attributes, but also in our case useful associations can be found. Formally, let \( F \) be a set of program features, and \( FI \) the set of all possible instances of the features in \( F \). Let \( D \) be the set of program ratings, where each rating \( Rp \) is uttered on a program \( P \) annotated by means of a set of instances \( PI \subset FI \). We can now say that a rating \( Rp \) contains \( X \), if \( X \) is a set of feature instances in \( FI \) such that \( X \subseteq PI \). An association rule is an implication of the form \( X \Rightarrow Y \) where \( X \subseteq FI, Y \subseteq FI \) and \( X \cap Y = \emptyset \). The rule \( X \Rightarrow Y \) holds in the rating set \( D \) with confidence \( c\% \) if \( c\% \) of the ratings in \( D \) that contain \( X \) also contain \( Y \). The rule \( X \Rightarrow Y \) has support \( s\% \) in the transaction set \( D \) if \( s\% \) of transactions in \( D \) contain \( X \cup Y \) [4]. In other words, the confidence of an association rule shows how frequently the rule head (\( Y \)) occurs among all the objects containing the rule body (\( X \)), or differently expressed: \( P(Y|X) \). The support of an association rule is the percentage of objects that contain all of the feature instances listed in that association rule (\( X \cup Y \)) or differently expressed: \( P(XY) \). The lift \( l \) of a rule is one of several important metrics to rank rules based on relevance. Lift expresses the importance of the rule, and can be defined by dividing the confidence of the rule by the support of the rule’s head (\( P(Y|X)/P(Y) \)). In other words, if the chance on having \( Y \) given \( X \) (\( P(Y|X) \)) is larger than the chance on having \( Y \), then the lift becomes larger than 1 which means that this rule is strong. To illustrate, if \( P(Y|X) \) equals 0.2 and \( P(Y) \) equals 0.1, than the lift becomes 0.2/0.1 = 2. In this case, the lift tells us that given \( X \), the chance on \( Y \) doubles. Association rules found by the Apriori algorithm, by applying it on our rating data set, include for example:

\[
(\text{channel = Eurosport 2}) \\
\Rightarrow \text{rating?='very low' support=11%, confidence=100%, lift=2.4} \\
(\text{day = tuesday}) \text{ and } (\text{dayframe = evening}) \text{ and } (\text{channel = RTL4}) \\
\Rightarrow \text{rating?='very high' support=4%, confidence=100%, lift=2.9} \\
(\text{day = saturday}) \text{ and } (\text{dayframe = evening}) \text{ and } (\text{genre = 3.2}) \\
\Rightarrow \text{rating?='very low' support=3%, confidence=100%, lift=2.4}
\]

The rules found confirm statements we previously found in the JRip (liking RTL4 on Tuesday evening) and Naive Bayes (not liking “Eurosport 2”) algorithms. If we look at the lift, we see that all three are strong important rules which definitely can contribute to the user model by materializing the correct liking assertions. We do however need to make sure that the support...
does not become too low. After all, it is possible to generate a high lift value while having a very low rule support. Therefore we use a configurable threshold (currently 10) to set the minimum support.

Conclusions

We can see that the different learners we evaluated here are able to perform reasonably. On average 60% of correctly classified instances is not bad, and about twice as good as our baseline prediction. We can think of various reasons for this problem. First of all, the main reason can be found in the quality of the metadata. This system mainly depends on the metadata of the programs it deals with. Besides all the enrichment steps we execute to make the metadata more elaborate, it can still happen that some program metadata remains poor. As a consequence it can for example be the case that a user rates one action program on channel X as very good, and another action program on the same channel as very bad. If the metadata is poor, then the system cannot differentiate between these too programs well enough, which in turn leads to more difficult classification learning. Secondly, the way in which a user rates can be confusing in itself. Users are not always consequent in their ratings which can lead to fuzzy statements in their profile. For example, users provide inconsistent ratings when asked to rate the same movie at different times [123]. Further, people also are more inclined to give feedback on incorrect predictions [242]. Thirdly, we previously saw that users tend to rate with extreme values ("very low" and "very high"). Therefore, we have very few programs rated 'low', 'medium' or 'high' which makes classification more difficult and leads to more incorrectly classified instances. However, given these issues, we are still convinced that we are able to detect some useful patterns in the user profile (given that we have enough instances) to augment the richness of the user model. Every data mining result is translated into a liking following Formula 6.2.3.

**Formula 6.2.3. Generation of likings from a data mining algorithm**

Let \( r \) be a resource, \( v_{dm} \) the value generated by a data mining algorithm for user \( u \) in context \( c \) and \( t \) the type of this algorithm. Then:

\[
L(u, r, c, t) = v_{dm}
\]

### 6.3 Cold Start

Systems which rely heavily on user information in order to provide their key functionality, usually suffer from the so-called cold start problem [221]. It basically describes the situation in which a system cannot perform its main functionality because of the lack of well-filled user models. This is not different in our approach. In order to make for example program recommendations, allow personalized search, etc., the system has to ‘know’ what the user’s interests are. In other words, when a new user subscribes to the system, it needs to try to discover as much user data as possible without bothering the user too much. Currently when registering to iFanzy, the user is asked to provide three features including his or her gender, age and education to obtain a first indication of what kind of person the system is dealing with. With these three key characteristics, more information is amassed in an unobtrusive way, by using three different strategies:

- Via historical viewing statistics: Users usually behave in predictable ways. Knowing their previous behavior provides a good clue for future behavior.
- Via import: Importing existing user data, by for example parsing an already existing profile of that user.
- Via classification: By classifying the user in a group from which already some information is known.
6.3. COLD START

<table>
<thead>
<tr>
<th>Channel</th>
<th>Date</th>
<th>Time</th>
<th>Genre</th>
<th>Duration</th>
<th>Total</th>
<th>20-24</th>
<th>W50-64</th>
<th>Scientific</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL1</td>
<td>01-04-08</td>
<td>19:02</td>
<td>Quiz</td>
<td>20</td>
<td>4.8</td>
<td>0.7</td>
<td>7.2</td>
<td>1.0</td>
</tr>
<tr>
<td>RTL7</td>
<td>03-11-08</td>
<td>21:27</td>
<td>Sport</td>
<td>48</td>
<td>2.6</td>
<td>4.2</td>
<td>1.0</td>
<td>2.3</td>
</tr>
<tr>
<td>NL1</td>
<td>01-07-08</td>
<td>20:00</td>
<td>News</td>
<td>25</td>
<td>9.3</td>
<td>1.8</td>
<td>14.7</td>
<td>9.8</td>
</tr>
</tbody>
</table>

Table 6.3: Entries in the historical program viewing statistics

All of these methods can potentially contribute to the retrieval of extra information describing the current user. In the following three sections we show how these methods are utilized to enrich the user model, which would allow us to even personalize the experience of new users to a reasonable extent.

6.3.1 User watching statistics

In most countries television viewing statistics are kept to produce for example lists of the most popular programs of the week. These statistics come from a select group of people spread demographically across the country, who share their viewing behavior with a central authority. Afterwards, these statistics are extrapolated to represent the entire population of the country. From all of the people in this group, several characteristics like age, gender, level of education, postcode, ethnic background, nationality, salary, social environment, family composition, etc. are known and used to introduce culture groups and stereotypes. An illustrative stereotype could for example be the group of people who are above forty years old, well-off, highly educated, having more than one child and living in Amsterdam. For such a group, we can then examine which programs on average were watched most, which is in turn interesting to predict their future preferences.

In iFanzy we currently have access to a list of all scheduled programs broadcast in the year 2008 (together about 33,000 entries), on the most important Dutch television channels. For every such entry we have the day and time of broadcast, a genre, the duration, the channel and the percentage of people who were watching this particular entry. Further, the list also divides this percentage over three key iFanzy characteristics: age, gender and education which are each subdivided in a number of groups. The age characteristic for example is divided into nine distinct groups: “6-8”, “9-12”, “13-15”, “16-19”, “20-24”, “25-34”, “35-49”, “50-64” and “65+”. Gender is obviously split into two groups and education has seven groups which represents the Dutch educational system. In the Table 6.3 we see some illustrative entries of this list. In the first entry we see for example a program broadcast on channel ‘NL1’ on the first of April 2008. This quiz had a duration of twenty minutes and 4.8% of all the Dutch people, who were watching television, watched this program. When we look now at more specific groups, we see that of all people with an age in the interval [20-24] only 0.7% watched this program. On the other hand, the percentage of women with an age in [50-64] who were watching, equals 7.2%. Similarly, the last column shows a percentage of 1.0% for people with a higher scientific education including master and doctoral degrees. In the second entry in Table 6.3, we see a sports program with an average percentage of 2.6%. In this case, we see that young people (4.2%) score considerably higher than older women (1.0%). In the last entry, we see that a news program is watched much more by highly educated people (9.8%) and older people (14.7%), than youngsters (1.8%). This information is extremely useful to us, because it clearly shows that on this specific time and day (context) one group of people was much more interested in a particular program than the average population.

The main goal here is to discover likings for groups of people, based on this historical data, which would help in predicting how a group feels about upcoming programs. This technique is very comparable to Collaborative Filtering, where live data from other users is used instead of historical statistics. Thanks to the availability of the time stamps in the statistics (through the date and time fields), we can deduce contextual information. From these two fields we extract time context in three categories: day frame (afternoon, evening, etc.), weekday (people tend to behave differently on different days of the week) and TV season. In the television world there are four so-called TV seasons in which a different listing of programs is scheduled. Hence, in
the winter season a different group of programs are broadcast than in the summer season. To exploit the information in this rather large table we again resort to data mining techniques. This time however, we are mainly interested in finding relations between fields. We want to discover for example that on average people over forty usually watch a quiz on Mondays. Or that males always watch football in the weekend and not or less on workdays. To find this kind of relations we again utilize the Apriori algorithm, already introduced in the previous section.

Previously, we ran our data mining algorithms in the Weka suite. However, Weka also introduces some constraints, especially with larger sets of data. The main problem is that it is difficult to tune and tweak the queries we want to fire in terms of confidence levels, rule support, inclusion of features, etc. Weka is not able to answer ad-hoc queries necessary in specific situations. For this reason, we considered inductive databases [135]. An inductive database tries to integrate the concepts of a regular database with data mining techniques. In an inductive database, not only data can be queried, but also patterns implicitly available in that data. Several systems implementing inductive databases have been proposed. However, in most of these systems they also introduce an inductive query language which extends an existing database query language, like SQL or XML, with some primitives to support data mining [73]. However, we would prefer the combination of a well-known query language like SQL with a data mining algorithm, which would enable some very interesting querying abilities. In [42], Blockeel et al. discuss a pragmatic approach to inductive databases by using virtual mining views. These mining views are relational tables that virtually contain the complete output of data mining algorithms executed over the data set. The prototype discussed by Blockeel et al. is implemented in PostgreSQL and currently integrates frequent item sets, association rules and decision tree mining. The main advantage of this system is the flexibility of ad-hoc querying where for example the user can specify new types of constraints and query the patterns and models in combination with the data itself. Furthermore, this system uses pure SQL as a querying languages without data mining specific additions.

To migrate the large viewing statistics table (as shown in Table 6.3) into a relational database structure, we first split the table into stereotype groups. Considering that we have 9 age groups, 2 gender groups and 7 educational groups, we obtain 126 (=9x2x7) possible combinations. However, we also want to add groups where for example age does not play a role (only the combination of education with gender) or where both gender and education do not play a role (only leaving the age groups). In this case we get 239 (=10x3x8)-1 possible stereotypes. From these we can then subtract 45 practically impossible groups like e.g. persons with an age less than 10 having a scientific degree, finally obtaining 194 realistic stereotypical groups. For each of these groups, we created a table in a PostgreSQL database extended with the mining views software, containing the 33,000 program entries.

Before we can migrate the historical viewing statistics to a relational table, one issue remains. In the mining views setup we want to utilize the Apriori algorithm, however, Apriori does not support numerical values. Therefore we first need to transform the percentages present in the statistics into classes. These classes should tell us how the percentage of viewers of a particular group, relates to the entire population. If 5% of the entire population watched program P and 10% of group G watched P, then the class expressing the interest of G in P should be ‘high’ or even “very high”. However, If 0.5% of the entire population watched program P and 1% of group G watched P, then the class expressing the interest of G in P should be rather ‘medium’, because, although the relative percentage increase was again 100% (from 0.5% to 1%), it still remains a small percentage (1%) of G who watched P. Therefore, our strategy differs based on the percentage of the entire population. In Table 6.4 we see the rules used for the conversion from percentage to a class. \( POP_p \) and \( G_p \) denote the percentage of viewing of the population and the group respectively on program \( P \). \( \overline{AVG}(POP) \) equals the average of the percentages of the entire population for all programs in the historical record in that particular TV season. For example, if \( \overline{AVG}(POP) = 3.5 \), \( POP_p = 4.8 \) and \( G_p = 7.2 \), then the designated class would be “very high” \((POP_p + 50\%(= 7.2) \leq G_p)\). If, on the other hand, \( \overline{AVG}(POP) = 3.5 \), \( POP_p = 2.6 \) and \( G_p = 1.0 \), then the designated class would be “very low” \((G_p \leq POP_p - 50\%(= 1.3))\). Considering again the group “women with an age in 50-64”, the respective relational database then contains among others the tuples shown in Table 6.5.
6.3. COLD START

<table>
<thead>
<tr>
<th>Precondition #1</th>
<th>Precondition #2</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{POP}_p &gt; \text{AVG}(\text{POP}) )</td>
<td>( \text{POP}_p - 20% &lt; G_p &lt; \text{POP}_p + 20% )</td>
<td>Medium</td>
</tr>
<tr>
<td>( \text{POP}_p - 50% &lt; G_p &lt; \text{POP}_p - 20% )</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>( \text{POP}_p + 20% &lt; G_p &lt; \text{POP}_p + 50% )</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>( G_p \leq \text{POP}_p - 50% )</td>
<td>Very Low</td>
<td></td>
</tr>
<tr>
<td>( \text{POP}_p + 50% \leq G_p )</td>
<td>Very High</td>
<td></td>
</tr>
<tr>
<td>( \text{POP}_p \leq \text{AVG}(\text{POP}) )</td>
<td>( \text{POP}_p - 20% &lt; G_p &lt; \text{POP}_p + 20% )</td>
<td>Medium</td>
</tr>
<tr>
<td>( \text{POP}_p - 50% &lt; G_p \leq \text{POP}_p - 20% )</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>( \text{POP}_p + 50% \leq G_p &lt; \text{POP}_p + 150% )</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>( G_p \leq \text{POP}_p - 50% )</td>
<td>Very Low</td>
<td></td>
</tr>
<tr>
<td>( \text{POP}_p + 150% \leq G_p )</td>
<td>Very High</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Rules to transform historical viewing percentages to classes

<table>
<thead>
<tr>
<th>Channel</th>
<th>Season</th>
<th>Day</th>
<th>Frame</th>
<th>Genre</th>
<th>Title</th>
<th>W50-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL1</td>
<td>Spring</td>
<td>Tue</td>
<td>Afternoon</td>
<td>Quiz</td>
<td>Lingo</td>
<td>Very High</td>
</tr>
<tr>
<td>RTL7</td>
<td>Autumn</td>
<td>Mon</td>
<td>Evening</td>
<td>Sport</td>
<td>Football</td>
<td>Very Low</td>
</tr>
<tr>
<td>NL1</td>
<td>Summer</td>
<td>Tue</td>
<td>Evening</td>
<td>News</td>
<td>Journaal 20u</td>
<td>Very High</td>
</tr>
</tbody>
</table>

Table 6.5: Entries in the historical program viewing statistics

Executing the Apriori algorithm in the, with mining views extended, database, resulted in among others the following rules (query is restricted to rules with a confidence of at least 90%):

\[
\text{(day = thursday) and (genre = Show) and (channel = SBS6) } \\
=> \text{ interest?=‘high’ with lift=12.8}
\]

\[
\text{(dayframe = afternoon) and (season = Spring) and (genre = Quiz) } \\
=> \text{ interest?=‘‘very high” with lift=6}
\]

The resulting rules (for the group “Women50-64”), with a high lift and a minimum confidence of 90%, can now directly be added to the stereotype model (a model containing all likings valid for, in this case, the group “Women50-64”) by means of liking assertions. The day, day frame and season fields in the rules are again used to materialize the context in which the rule is valid. Lastly, from these statistics we can also derive potential liking not only on programs but also on channels and genres, as explained in Section 6.2.3.

6.3.2 Community profiles

Looking at the evolution and growth of Web 2.0 social networks like Hyves\(^3\), Facebook\(^4\), LinkedIn\(^5\), Netlog\(^6\), etc. we must conclude that users put a lot of effort into building an extensive online profile. Often, networks grow within a country (or language group) to become dominant while remaining much less known abroad. Hyves for example is a huge hit in the Netherlands, while almost not known outside. In Figure 6.13 we see an overview of social network usage across the world according to language areas. In this figure it appears that Facebook is by far the dominant global network. Looking at these numbers, it is no surprise that there has been a lot of effort in devising ways to benefit from this huge amount of user data. Facebook, with more than 400 million active users\(^7\) in June 2010, started with the introduction of the Facebook platform (a set

\(^3\)http://www.hyves.nl/
\(^4\)http://www.facebook.com/
\(^5\)http://www.linkedin.com/
\(^6\)http://www.netlog.com/
of APIs) in May 2007 which made it easy to develop software and new features making use of this user data. Afterwards others followed, and in response Google started (together with MySpace and some other social networks) the OpenSocial initiative⁸, which is basically a set of API’s which makes applications interoperable with all social networks supporting the standard. In Figure 6.14 we see the growth of OpenSocial compatible networks in terms of number of users.

![Image](http://code.google.com/apis/opensocial/)

---

⁸http://code.google.com/apis/opensocial/
In iFanzy, the Hyves network was particularly interesting as a source of user data. The choice was straightforward considering that Hyves is by far the biggest network in the Netherlands with over 7.5 million users (which makes almost 50% of the population). Furthermore, Hyves is also a partner of the OpenSocial initiative. For iFanzy, we are specifically looking for interests defined by the Hyves users, because among them there can be interests in programs, people, locations, etc., which could be useful to fill the user model at cold start time. Let us take a look at the Hyves profile of an average Dutch person called ‘John’. In Figure 6.15 we see an overview of such a profile. Depicted we see an overview of his Hyves interests classified according to the Hyves categories. Among these categories there are (translated): movies, music, traveling, media, tv, books, sports, food, heroes, brands, etc.; most of which are interesting for us to retrieve.

Unfortunately there were some shortcomings to the available Hyves API’s. First of all, via the regular Hyves API, it is not possible to access all fields of the profile (including the interests shown above). The API only allows you to retrieve simple values like age, birthday, friendscount, etc. (for more information see http://trac.hyves-api.nl/hyves-api/wiki/APIMethods/1.0). Therefore, we decided to build a custom HTML parser for the Hyves profile page. Given the username and password of a Hyves account, and therefore the user’s consent, our crawler parses and filters the most interesting values of the user’s profile page (currently only available in the iFanzy test environment).

| Music: | 50 Cent, Acca & De Munnik, Air, Akon, Alicia Keys, Aphex Twin, Blof, Bob Marley, Cypress Hill, Don Diablo, Dr Dre, Ella Fitzgerald, Erick E, Faithless, Guus Meeuwis, Mary J Blige, Orbital, The Prodigy |
| Reizen: | Afrika, Amsterdam, Carribbean, Cuba, New York, Parijs, Strand, Wintersport |
| Overig: | Auto’s, Technologie |
| Media: | BNN, Computer Totaal, hardware info, MTV, Net 5, RTL 4, RTL 5, RTL 7, SBS 6, The Box, TMF |
| Gadgets: | Creative Zen |
| TV: | De Lama’s |
| Boeken: | Getting Things Done, iets, Lord of the Rings |
| Sport: | Hardlopen, Snowboarden, Tennis |
| Eten: | pizza, Sate |
| Helden: | Roger Federer, Al Pacino |
| Merken: | Asics, Brooks, Calvin Klein, Diesel, EDC, Esprit, G-star, Gilette, Hugo Boss, Jupiter, Lacoste, Le Coq Sportif, Mexx, Nike, Nivea, Oakley, Puma, Replay, Sony, Tommy Hilfiger |

Figure 6.15: A relevant part of John’s Hyves profile

In order to import the Hyves interests in a user model, we run a matcher which tries to match the interest strings to a concept in our RDF/OWL graph. Depending on which category of interest, a slightly different approach of matching is applied. In the categories ‘movies’ and ‘tv’ we try to find matches within our set of TV programs and persons possibly involved in those programs. As Hyves does not enforce any restrictions on what you enter in a certain category, there is no guarantee on the percentage we can match correctly. In the ‘media’ category people can put interests in all kinds of media objects like newspapers, tv channels, magazines, etc. Our matcher compares all these strings to all objects representing channels and streams. In this example, ‘MTV’, “Net 5”, “RTL 4”, etc. are all matched to the respective television channels. The same tactics are applied on the other relevant categories, and thus we match ‘traveling’ (e.g. ‘afrika’, ‘amsterdam’, ‘cuba’, etc.) to geographical locations, ‘sport’ (e.g. ‘tennis’) to our genre hierarchies and ‘heroes’ (e.g. “Roger Federer”) to our list of persons.
Once the Hyves interests are matched to a certain concept in our database, we have to materialize these relations into the user model by making liking assertions. For example, the Hyves strings Hyves.Films.Scarface, Hyves.Reizen."New York" and Hyves.Helden."Al Pacino" are matched to IFZ:scarface1983 (the movie), Geo:5128581 (the Geonames object for 'New York') and IFZ:alPacino1940 (the person "Al Pacino") respectively. While we illustrated the functionality by means of the Hyves network, many other already existing user profiles of different social networks could also qualify for this approach.

6.3.3 User classifications

Next to trying to retrieve and use social network data to gain user information, we also acquire extra knowledge by classifying the user in specific groups from which we already have some information available. When we know that members of a specific group of people have a similar opinion about a group of program items, and the new user is similar to these group members, he will most likely also share the same opinion on those items. This approach is also known as Collaborative Filtering (first mentioned in [107]) and already widely accepted by numerous commercial systems like Amazon, eBay, iTunes, Last.fm, Netflix and TiVo. However, in order to be able to perform a collaborative filtering algorithm, the system needs at least a reasonable group of users which all gave a reasonable number of ratings. Secondly, collaborative filtering is truly valuable when dealing with a more or less stable set of items like for example a list of movies. This is due to the “first rater” problem. When a new item enters the item set, it takes time before it receives a considerable number of ratings, and thus it also takes time before it is known how the different groups feel about this item. This is in particular a problem in the TV world, since new programs (or new episodes of series) emerge constantly, making this a very rapidly evolving data set.

Since iFanzy is still a beta project, the available active user base is still too small to be able to perform any kind of collaborative filtering strategy. Therefore, until the user base reaches a size that allows us to apply the desired collaborative filtering technique, external group classifications are used to guess how a person might feel about a certain item. External data sets currently used include the MovieLens rating set and the IMDb ratings classified by demographics.

MovieLens Classification

MovieLens is a, collaborative filtering based, movie recommendation system built by the Grouplens Research group at the university of Minnesota [122]. Via their Web site they encouraged people to rate movies until they amassed a data set consisting of thousands of users and millions or uttered ratings. In this data set, they maintain a tuple for every movie <mID, Title, Genres> and for every user <uID, Gender, Age, Occupation, Zip-code>.

Based on gender and seven age groups we can, from this information, generate fourteen groups in which we can classify iFanzy users. For each of those groups we calculated the average rating for all the items in the MovieLens item set, where a minimum number of ratings for each item was required. Having all users in iFanzy classified in such a group, we can predict how much they would appreciate a specific item (if it exists in the item set). The disadvantage of the MovieLens data set is that it only contains ratings for movies and thus no regular television programs, which are still the most common items. However, by introducing this data in our data graph, we can propagate these ratings on the genres connected to these movies, such that the benefit reflects on other programs as well.

IMDb Classification

IMDb keeps, besides a general rating for all of its items, also all the independent ratings of the people, spread over their demographics, who uttered them. Besides gender, they also split the rating data into four age groups, combined resulting in eight groups in total. Just like with the

9More can be found on http://en.wikipedia.org/wiki/Collaborative_filtering#In_commercial_systems
10http://www.grouplens.org/
data of MovieLens, we classify our users in these eight IMDb user groups. To show how significant these differences between groups can be, let us take a look at the movie ‘Scarface’, which has a general IMDb rating of 8.1/10. We see that on average, males under 18 years give a rating of 8.7/10 while females over 45 years rate this movie 5.7/10. Like this example clearly shows, it pays off to classify users based on their demographics.

While the MovieLens ratings data set has a higher granularity with respect to age distribution, the IMDb set has a clear advantage considering data coverage. IMDb does not only have ratings on movies, but also on television series and various other shows. In general, we can say that these two practical classification methods are very effective in the current situation where iFanzy’s user base remains limited (from the perspective of collaborative filtering). By classifying the user in both IMDb and MovieLens defined groups, we immediately unlock all available ratings in those sources which in turn are used to kick-start the user model of a new user. Once the user starts uttering ratings himself, they slowly replace the classification ratings. Once more and more users rate programs, we start applying collaborative filtering techniques, exploiting similarities between persons to form groups on the one hand and between these groups and TV programs on the other.

6.4 Conclusion

In this chapter, we discussed the user modeling component introduced in the approach proposed in Chapter 4. There we saw that making any data retrieval or consumption more personal, requires a good comprehension of the user’s relevant characteristics, preferences and interests. Moreover, this user information needs to be saved, structured and filtered before it can be interpreted by any automated process. In this chapter we explained our user model approach in support of enabling user-adapted data retrieval.

Further, we discussed a flexible RDF(S)/OWL based user model which is able to accommodate various user aspects, characteristics and preferences. Furthermore, all user feedback and/or statements are stored as assertions modeling the user’s reflection or viewpoint on the central domain items (here including resources like e.g. programs, persons, genres, channels, etc.). We have also shown that this user model is equipped to model user context, elucidating in which situation a specific assertion is valid. For every fact which we elicit, a reasonable amount of contextual user information is gathered, by carefully inspecting the user’s behavior and the actions performed. With the creation of this model in all of its aspects, we answered research question 5A (How can we model relevant user data to support context-sensitive adaptation?).

Within this user model we introduced the concept of user events. These user events are the carriers of all user actions, performed both explicitly and implicitly, and accompanied by contextual information. This constant stream of user-generated events is stored and afterwards mined to extract valuable patterns. From these patterns, information can be extracted which in turn can lead to the materialization of facts in the user model by means of likings. In this way we can amass and utilize every bit of information the user leaves behind when interacting with the system. This process therefore provides an answer to research question 5B (How can we obtain this user data encompassing both explicit and implicit data?).

This user modeling approach was inspired by the idea that we should get as much information from the user as possible in an unobtrusive way. This is specifically important at cold start time, where we need all the information we can get to kick-start the user model. After all, we realize that an initial lack of information is inevitable when a new user subscribes. However even at cold start time, we want to be able to provide an, albeit less refined, personalized experience from the start. Through careful domain analysis we concluded that having the user’s gender, age and education already provides good clues in predicting what he or she might appreciate. In this chapter, we have described three techniques which help at cold start time, including learning from historical viewing statistics via data mining techniques, importing and parsing already existing user profiles (in this case from Hyves) and classifying users into groups which are already described on the Web. Therefore, this section provided an answer to research question 5C (How can we support new users who suffer from an empty user model?).
CHAPTER 6. USER MODELING

6.5 Related Work

Since the late 90’s there have been a number of systems developed that incorporate different approaches to collecting user preferences (e.g. implicit or explicit) and various filtering strategies. For example, [70] employs explicit techniques for the users to indicate their TV preferences and interests. The main disadvantage of this approach is that it requires an active user’s involvement and being quite static, since it does not allow for evolution of user profiles over time. Similarly in [68], Cotter et al. use explicit user preferences about channel and genre as well as explicit user feedback on watched TV programs, and infers the preferences for other programs. In [10], Ardissono et al. use a hybrid approach to content-based recommendations by combining explicit, stereotypical and implicit user preferences, which is similar to what we do in iFanzy. However, in our case both implicit and explicit feedback contributes to a consolidated value expressing the user’s contextualized interest in specific concepts in the semantic data graph. Typically, other recommendation systems also do not incorporate or use limited contextual user information, partially because it remains a difficult task to elicit this information from the user. However, in an e-commerce application like the one presented in [7], Alshabib et al. propose a recommendation model that combines context-based ratings with the structure of a social network. The context of a user rating considered here defines a particular category within which a service is being rated (such as Movies, Book:Fiction, etc.). In our approach, we consider a temporal context (e.g. the time/day of the week of the TV broadcast and the current time of the user) and an action-based context (e.g. monitoring the user’s TV viewing events, such as “program recorded”, “added to favorites”, “recommended to a friend”). Further, in [7] a social network (LinkedIn) is used to aggregate ratings based on the structure of the network, by calculating the neighborhood of users. We take a similar approach in this paper, by using a different social network (Hyves) to find additional user interests. We also incorporate various background knowledge (e.g. semantic relations between genres, demographic statistics about TV channels and the user’s age and gender) which creates a more interconnected data graph, allowing us to relate various concepts, and as a result concept ratings.

[261, 258, 237] argue that not taking situational information into account for recommendations, seriously limits the relevance of the results, and as in iFanzy, they advocate context-awareness as a promising approach to enhance the performance of recommenders. While [261] and [237] illustrate their framework with a restaurant recommender, taking location, weather and restaurant data into account, in [258] Woerndl et al. apply context-aware recommender systems (by using location and acceleration) in the domain of inter-networked cars. In [261], Yap et al. separate contextual concerns from recommendation, so that contexts can be readily shared across applications.

In [58], Byun et al. propose a context-aware Personal Digital Secretary (PDS), which is responsible to find the relationship between context and user model information. The PDS includes sensors and various interfaces to discover these kinds of relations. iFanzy could benefit from an extension in these directions too.

In [196], Rack et al. study the synergy between user behavior, context data, and semantic information in order to enable services to adapt to different situations based on the recommendations of a service independent recommender. Just like iFanzy, AMAYA delivers similar context-aware recommendations which are based on provided feedback, context data, and an ontology-based content categorization scheme. While we model a profile as a set of assertions, in [196] Rack et al. call a single unit of personalization data an ‘entry’ of a profile. However, the authors elaborate less on what the context there specifically covers. Like in iFanzy, they also maintain an ontology-based content categorization scheme containing concepts like e.g. “sport news” and “business news” as well as relations like e.g. “soccer news is a sub-category of sport news”. However, they do not consider general ontologies, e.g. to model time or geographic locations, as we do in iFanzy.

As previously seen in this chapter, in every system where the user model is essential, the cold start problem arises. This is usually the case when a new user enters the system. While we chose an unobtrusive way to elicit as much information as possible, others make use of more apparent methods. For example in [129], Hsu et al. solve this problem by requesting its users to fill out a questionnaire when they register at the TV program recommender. The questionnaire
is to gain knowledge about demographic data, interests and activities, and program category preferences of its users. Afterwards, this data is used to classify users into viewing groups from which already some information about their interests is known. iFanzy also deploys demographic filtering techniques, however, users first have to provide age, gender and education, to classify them accordingly. In [194], users have to define a profile for themselves at the beginning, before a learning algorithm takes over based on explicit user ratings.

Just like presented here, others also deployed data mining algorithms to for example learn user model patterns for television recommendation systems. In [194], Pronk et al. propose an extension to the naive Bayesian classifier to enhance user control. By maintaining and integrating two profiles for a user, one learned by rating feedback, and one created by the user, they try to mitigate the cold start problem. As more patterns are learned through user ratings feedback, the learned model gradually takes over the user provided model. In iFanzy we do not let the user create a profile themselves, rather we mine their implicit feedback events. In [188], O'Sullivan et al. use a collaborative filtering algorithm to recommend users a set of programs fitting their taste. However, the collaborative filtering approaches suffer significantly from the sparsity problem, which exists because the expected item-overlap between profiles is usually very low. As a solution, they address the sparsity problem by proposing the use of data mining techniques. By using the Apriori association rule mining algorithm, they try to discover new knowledge indicating similarities between user profiles.

Just like we did, [32], [239] and [118] also recommend using a generic user model in personalization systems. [32] proposes a general framework and specific methodologies for enhancing the accuracy of user modeling in recommender systems by importing and integrating data collected by other recommender systems. [118] focuses on the creation of GUMO, a general user model ontology for the uniform interpretation of, and communication between distributed user models. GUMO, which is modeled in OWL, particularly targets intelligent Semantic Web enriched environments. Further in [239], Van der Sluijs et al. use Semantic Web technology to support the exchange of user data between applications. The presented the generic user model component (GUC), which provides user model storage facilities for applications. In iFanzy we also developed a model keeping in mind that it could also serve different domains and should be able to be used by third-party applications. However, due to lack of standards for the representation of user models, privacy issues and commercial competition, application builders usually develop their own specific user models and store their information in incompatible manners [31].

Lastly, as shortly introduced in Chapter 2, TV-Anytime defines a standard way to describe consumer profiles including search preferences and usage history. As described in ETSI TS 102 822-6-3, the TV-Anytime specification describes a user profile service that acts upon some resource to either retrieve and update information (such as preferences, action history, demography, etc.) about a user. According to ETSI TS 102 822-3-1, user data is saved by means of the MPEG-7 MPEG7:UserPreferencesType and MPEG7:UsageHistoryType complex types. However, in comparison to the user model defined in this chapter, these constructs are quite limited and inadequate with respect to the requirements listed in Chapter 3 in terms of context and flexibility.
Chapter 7

User-Adapted Data Access

According to the Oxford dictionaries, the definition of personalization is stated as following: “design or produce (something) to meet someone’s individual requirements.”. In other words, taking a general statement and tweaking or adapting it in such a way that it becomes specifically tailored for a particular individual. For example, when John goes to the bakery to buy a loaf of bread, his good friend Jim, who works there, automatically takes the special raisin bread and cuts it into slices much larger than usual, because he knows that John likes it that way. To perform this act, Jim needs to know two things: 1) the properties of every specific type of bread as well as how to set the cutting machine accordingly and 2) John’s preferences and particularly that he likes the raisin bread and moreover, he likes it best when cut in slices of exactly 15 millimeter. The combination of these two distinct sets of information allow Jim to perform the action that makes John a content customer, namely taking the right bread and cutting it correctly.

However, personalization is far from an absolute concept and knowing what and how a person should be served at any given time, can at best only be approximated. After all, the choices people make in their daily life are driven by numerous personal and external influences, therefore making them potentially unpredictable and hard to grasp. The fact that John found himself a new girlfriend last evening might suddenly make him longing for a sugar bread instead of the usual, or just seeing an advertisement of some chocolate brand right before he entered the bakery, can lead to ordering a nice piece of chocolate pie. Changes of mind, mood, feelings and external stimuli as well as their subtle propagating effects are currently far from predictable, as the human brain remains the biggest enigma of modern science. Still and luckily for us, in most situations people like to follow daily patterns guiding their behavior and lifestyle, as thoroughly studied in sociological sciences like Ethnomethodology [103]. Being the attentive baker that Jim is, he for example quickly notices that John looks particulary happy today, therefore asking him whether he would prefer a sugar loaf instead. Modeling and capturing the user’s preferences and interests is one thing, making the most out of them is another, and furthermore depends on that modeling.

In this chapter, we propose a strategy which personalizes the process of data retrieval. The goal of this strategy is to retrieve exactly that set of items fitting the user best, given the current data set (as defined in Chapter 5), the contextual situation and the user model describing that user (as defined in Chapter 6). This strategy provides an answer to research question 6 (How can we provide user-adapted data access given a well-defined domain model and a comprehensive user model?).
CHAPTER 7. USER-ADAPTED DATA ACCESS

7.1 Introduction

Personalization is the often provided answer to the question of how we can deal with enormous amounts of data, independent of the domain, to which users are granted access [146, 207, 10]. Furthermore, personalization is becoming increasingly important, largely driven by the growth of information repositories and Web page complexity. The term ‘personalization’ should not be confused with Adaptation which is another related and well-known research topic. In the literature, several related terms have popped-up and several definitions have been formulated [120, 5]. Adaptation is a more general term encompassing various kinds of customizations, whereas personalization only deals with user-centered adaptation. Environments portraying adaptation in general, comprehend systems which are adaptable and/or adaptive. Adaptable or customizable systems allow the user to tweak specific aspects of the interaction, presentation, navigation, etc. of the system himself (explicit). Adaptive or personalized systems actively monitor and study the user’s behavior in an effort to alter certain aspects of the system fitting that particular user (implicit). Personalization can manifest itself in potentially every aspect of the data flow. It can lead to for example personal search, personal recommendations, filtering of inappropriate items, personal navigation through the data, personalization of the presentation, etc. Therefore, personalization requires some user information to be able to adapt specific functionalities like e.g. searching, navigation, presentation, etc., to that person.

In the real world however, things are usually not personalized. In case of searching for example, when both my ten years old nephew and my professor enter the same query in an average search engine, the results will be the same although the information need is quite different. Therefore, from the user’s perspective, the difference between a data source returning millions of results vaguely matching the query (like most search engines do) or one just returning a few hundred personal matches, is huge. We can only imagine how a personalized Web would affect general user satisfaction.

Regrettably, most search engines (or other Web services) do not provide personalization and their endless lists of results do contain a lot of irrelevant information, making the user himself responsible to continue his search, through these lists, manually. Partially this is due to the fact that user queries are so short, that search engines generally do not receive enough detail about the user’s information need, resulting in many irrelevant results [105]. However, we are convinced that user satisfaction would rise considerably in case only a limited set of results is returned, but where every one of those fits the user perfectly. In many occasions we can already estimate whether or not the user will appreciate a specific item pretty good, without involving the user explicitly. It just requires some user knowledge. E.g. if the user searches for the word ‘knight’ and we know that he does not like the genres ‘Thriller’ and ‘Horror’, there is no point in returning the movie “The Dark Knight” which is annotated with ‘Thriller’. And exactly this “sharing of personal information” is a sensitive point. It is a thin line between increasing the functionality’s quality making the user happy and asking for access to more personal user information usually making users less cheerful. However, previously we have shown that we can go a long way with relatively few user information. Therefore, we strive for profound personalization methods without requiring too much user data, still enabling for example personalized search, recommendations, information presentations, etc.

Following our general approach presented in Chapter 4, to facilitate this kind of functionality two requirements need to be met. Firstly, we need to have a thorough understanding of the current domain and its relevant items. E.g. working in the literature domain requires, at least, a good understanding of books and magazines as well as profound metadata descriptions for those book instances, before we can decide which book can be recommend best. The same holds for the television domain. If the thriller “The Dark Knight” is not annotated with the genre ‘Thriller’, we cannot remove it from the results even though we do know that the user does not like that genre. Secondly, understanding the item’s properties is only relevant if we also know the user’s perspective on that item. If the “The Dark Knight” is annotated with the genre ‘Thriller’, and we do not know that the user does not like programs annotated with this genre, there is no ground to remove this program from the search results. Considering that these two points were discussed in
great detail in Chapter 5 and Chapter 6 respectively, we have obtained all necessary ingredients to devise a strategy personalizing the user's experience.

Although many different forms of personalization exist in various parts of the data flow, due to our illustration by means of the television domain we will focus primarily on searching and providing recommendations for TV programs. Personalizing data access basically means that every user obtains a personal view of the available data. In other words, when two different users form the same request in a client application running on this framework, each one will obtain a custom result constructed based on their individual user model containing their interests and preferences. To obtain such personalization, there are basically two approaches:

- **Query-only approach**: In this approach we analyze every user query and reconstruct it taking the user model into account, forming a personalized database query. We then ask the database to answer this updated query, restricting the results to items which the user definitely will appreciate. E.g. if the user does not like the genre 'Horror', the query would contain a where-clause excluding items annotated with that particular genre.

- **Query+Filter approach**: Here, the query is sent to the database untouched. However, when the database returns its results, a filtering step is executed which filters these results based on the user model. E.g. if the user does not like 'Horror', the results from the original query (possibly containing items annotated with the 'Horror' genre) are filtered excluding items annotated with that particular genre.

These two approaches basically offer the choice between applying user model information to the query or afterwards in the result set. Both of them however have pros and cons. The query-only approach’s main advantage is that the query result immediately provides the complete personalized answer to the initial query. However, its main disadvantage is that the query might become very large, which makes it dependent on the database’s query optimization engine to efficiently deal with the various restrictions. Moreover, as the list of user model preferences grows, the query grows proportionally. In the query+filter approach on the other hand, the query is not depending on the user model, which means that the query’s complexity stays constant. Instead, a filtering process personalizes the query result based on the user model. A disadvantage here is that we might be facing large numbers of results coming from the database which all need to be dealt with by the filtering process. On the positive side however, filtering the results at the end gives more flexibility on the filtering process. The system can in this case for example determine a specific order in which to execute different filters, or new filters could be added dynamically (in a format understood by the filtering process) while the system is running. In any way, we believe that the system can react more flexibly in any kind of situation when the personalization is largely controlled by the system itself.

Considering the pros and cons of these approaches, actually a combination of the two seems like the most flexible approach. Here the system could try to combine the best of both worlds by optionally adapting the query to ‘steer’ it in a particular direction, while afterwards, a filtering step is executed to further personalize the results. By slightly steering the query adaptation, we keep control over the potentially growing complexity of the query, as will be explained further in this chapter.

In the following section we explain our personalization process, which contains four important parts, namely a **Query Refinement** component, the **Database management system**, a **Content Filtering** component and the **Recommendation** system. We discuss each of these four components, and explain how each one contributes in making data access personal. Although the process is explained with examples from the television domain, the approach itself is domain-independent.
7.2 A personalized data access strategy

The personalization strategy which we present here is a stepwise process starting with a user query and ending with returning a set of results, which are specifically tailored according to a user model, to that user. In Figure 7.1 we see an overview of the entire process. On the top left, the user uses a particular interface to make a request like for example: “Give me all sport programs between 3pm and 11pm on Dutch channels”. After receiving the request, the Query Refinement component interprets it and transforms it into an extended query which is sent to the Query Handler. The Query Handler parses the extended query, and sends it to the Database Management System (containing all relevant databases) which eventually returns a set of programs matching the constraints (in this case that the programs must be sport programs and be broadcast between 3pm and 11pm on Dutch channels). This resulting set of programs is then sent to the Content Filtering component which applies a set of filters (e.g. filtering inappropriate programs out), and then assigns a ranking value. Once filtered and ranked, the results are returned to the user.

![Figure 7.1: The personalization process](image)

In the middle of Figure 7.1, we see three data repositories together with the communication channels (dashed lines) between them and other components. These three data repositories each have a specific focus and provide necessary knowledge supporting the here described personalization process. Furthermore, their knowledge is domain-independent. Briefly describing them:

- **User Model Data**: The first repository contains all user model data of every user in the system, together with the ontological representation of the user model structure. It catches generated events, deduces new information, counters cold start problems and deals with the user’s context as described in Chapter 6.

- **Ontological Knowledge**: The second repository contains all relevant ontological background information, combined with domain knowledge as described in Chapter 5. In this repository
7.2. A PERSONALIZED DATA ACCESS STRATEGY

we store all the ontologies, vocabularies and thesauri which are used during the personalization process and relevant in the current domain. Every time we need knowledge of concepts, properties or hierarchies, we can query it using specialized API calls. We can for example ask for all synonyms of the word ‘bike’, which would return among others the word ‘bicycle’. We can also for example ask for the range (via \texttt{rdfs:range}) of a property to interpret its value or all subclasses of the class \texttt{Person}. The repository further allows for the conceptualization of a keyword via any of several annotation properties like \texttt{rdfs:label}, \texttt{skos:altLabel}, \texttt{skos:prefLabel}, \texttt{wn:lexicalForm}, etc. Conceptualizing the word ‘evening’ would return a \texttt{time:ProperInterval} instance describing the concept evening starting at 6pm and ending at 11pm in the OWL Time ontology, but also the corresponding instance of the WordNet class \texttt{wn:Word}.

- Filters and Rules: The third and last repository is responsible to maintain a set of rules which can be used to filter or rank results in the query result set. This repository allows for rules to be added dynamically, while the system is running. In these rules, references to ontologies can occur to refer for example to the user’s age field, the genre of a program, etc. Hence, this also explains why relations exist between this repository and the previous two.

As shown in Figure 7.1, most components make extensive use of these three data repositories, and we will explain these connections more detailed later on. The Query Refinement component for example uses the user model in combination with ontological information, while the Content Filtering component uses all three repositories.

7.2.1 Query Refinement

In most search engines, users are required to enter some kind of request by means of a keyword entry field. Just type some strategically chosen words, and matching results appear. Unfortunately, keywords can be very unprecise and sometimes even ambiguous. Still, for the majority of the people this is a very natural way to search through large information sources like the Internet. However, a minority of these users do understand the subtle pitfalls of searching, and could therefore be supported greatly with more advanced search methods. By for example allowing users to search for one or more exact concept(s), defined by a URI, in addition or in combination with a regular keyword search, knowledgeable people could indicate very well what exactly they are looking for.

The Query Refinement component is responsible to extend and enrich the original user request. It can deal with three different kinds of input: a basic keyword string (e.g. “sports program this evening”), a list of ontological concepts (e.g. “genre:sports, time:evening”) or a combination of the former two (e.g. “genre:sports this evening”). Do note that these three different kinds of input is what our approach allows. What the effective user interface allows the user to do or how he or she can form a query there, is not relevant for this discussion. Generally, we can present such a query like \([K_0, \ldots, K_p, C_0, \ldots, C_q]\), which is the conjunction of \(p\) keywords in the keyword list \(K\) and \(q\) concepts in a concept list \(C\).

To parse queries of this format and turn them into a more refined and personal query, we provide an approach consisting of three consecutive steps. This approach has a determined focus on conceptualizing keywords in the query, since searching for a concept defined by a concrete URI is more exact, and can be executed faster, than searching for one or more keyword(s). Therefore, the first step of our approach removes stop words and tries to find the stem of every remaining keyword to simplify the keyword set \(K\). Secondly, we try to conceptualize the remaining keywords in an effort to make the query as concrete as possible. In other words, we try to find the concept(s) which exactly represent every keyword. However in this conceptualization step, many concepts can potentially be found. The Ontological Knowledge repository just returns all the concepts which match with the given keyword. Moreover, due to a clear separation of concerns, this makes it a user-independent process, meaning that the user model is not involved in this process at all. Consequently, after the conceptualization step where we obtained a set of matching concepts, we
have a concept selection step which trims this set taking among others the user model into account. Here, we basically want to select exactly those concepts which describe the given keywords best and are most relevant for this particular user. In the following three paragraphs we further illustrate each of these distinct steps more thoroughly:

- **Keywords parsing**: Keywords are basically parsed in two ways (where both are optional). Firstly we filter out all stop words by means of a Lucene stop word filter\(^1\). Secondly, we use the Porter Stemmer\(^2\) as described in [191] to discover and add the stem of keywords (e.g. ‘sporting’ leads to ‘sport’). Both the stop word filter and the stemmer are instantiated with the correct language file (deduced from the keywords themselves if possible, otherwise the native tongue is chosen from the user model). The resulting query’s keyword set \(K\) after these two steps, is therefore updated by removing keywords which are filtered by the stop word filter and adding new keywords which resulted from the stemming process.

- **Keyword conceptualization**: All keywords left in \(K\) after the previous step, we try to conceptualize. Conceptualization here means that we try to discover the semantics of these keywords. To do so, we make use of the Ontology Knowledge repository. There we basically check, in a user-independent way, whether we can find the keyword in any of the labeling properties of the classes in all the ontologies, vocabularies and thesauri maintained there. This matching algorithm has three modes, i.e. strict, loose and free. In the strict match the keyword should be syntactically equal to the label (ignoring case), e.g. the keyword ‘news’ and the genre concept IFZ:News match. Loose match looks for substrings, e.g. the keyword ‘news’ and the genre concept IFZ:Daily News match, where it would not match in strict mode. Free match does pattern matching and would find the term ‘day’ as a match with TIME:Friday, where it would not match in loose match. The output is a set of concepts where it remains possible that one keyword matched several concepts. Do note that keyword conceptualization can be escaped by surrounding a keyword (or sequence of keywords) with quotes. E.g. a user searching for the movie “Last Action Hero” can in this way prevent the keyword ‘Action’ from being conceptualized to the concept describing the genre ‘Action’.

- **Concept selection**: After keyword conceptualization, \(x\) keywords resulted in a set of \(y\) concepts. However among these concepts, it occurs that less relevant or inappropriate concepts are included. Therefore, this set of concepts is not just added to the query. Rather, we first execute a Concept Selection step which filters this set of concepts to only keep the most relevant ones. The automatic selection removes concepts in two ways:
  
  - **User model filtering**: By looking at the user model, some concepts can immediately be removed from the set. For example, concepts, for which the user previously indicated that he or she does not like them at all, are removed. This can either be indicated by the fact that the user rated the concept directly (e.g. the genre ‘Action’ has a low rating), or by a deduced liking (e.g. the majority of the programs with genre ‘Action’ have a low rating).
  
  - **Concept Disambiguation**: Often, the conceptualization step discovers several concepts for one particular keyword. E.g. the keyword ‘evening’ can result in the concepts TIME:Evening and WN:Evening (the WordNet entry modeling the word itself). By prioritizing concepts, we can remove the WordNet contribution because it is more likely (although not certain) that the user is searching for programs in the evening. In general, we prioritize all knowledge structures (ontologies, thesauri, etc.) in such a way that the most relevant ones (mostly the domain specific structures) end up with a higher priority than the more general ones. Hence, if a concept from a specific domain ontology perfectly matches the keyword, it has a higher chance of being the right conceptualization in comparison to a concept from a very general domain-independent structure.

\(^1\)http://lucene.apache.org/  
\(^2\)http://tartarus.org/ martin/PorterStemmer/
In an advanced non-standard setting, it is possible to involve the user in this selection process. After the conceptualization process, we start a dialogue with the user where he or she can select the exactly intended conceptualization out of all different matches found. Eventually, the Concept Selection component returns the filtered set of concepts which will be used to update the query, by adding them to $C$.

Following these three steps, the Query Refinement component is able to transform a user request, which can both consist of keywords and concrete concepts, into a more concrete and personal query. The process tries to filter the query keywords, guess intended meaning of words and disambiguate concepts. This query refinement procedure is the first step of our user-adapted strategy, personalizing information access. Therefore, the same query will obtain different results for different users. The result of the query refinement process is sent to the query handler which further processes it (like further explained in Chapter 8) before sending to the Database Management System.

### 7.2.2 Database Management System

After having obtained the final set of concepts and keywords that conveys the user's request, the Query Handler constructs a set of queries which are sent to the Database Management System to fetch matching program descriptions. However, we do not only want to retrieve exact matches, but also highly related or other potentially interesting programs. In this way we can provide the user with a broader result which is still relevant with respect to the initial query. It is here where the use of rich metadata in combination with Semantic Web techniques really shows. The Database Management System, which maintains all the metadata following the structure presented in Chapter 5, can exploit well-defined semantic relations between concepts to broaden the search space in a controllable fashion. To do so, the database system is fitted with a reasoner. A reasoning engine is a piece of software, in our case active within the database, able to infer logical consequences from a set of facts. When provided with a set of custom rules, working on top of some carefully selected semantic relations in the ontological graph, relations can be deduced which are taken into account when a database query is parsed. We illustrate this functionality here briefly by means of two concrete examples.

Firstly, in Chapter 5 we saw that we used the SKOS relations `SKOS:broaderTransitive` and `SKOS:narrowerTransitive` to model the hierarchy of TV program genres. These properties indicate that the relation is transitive, which in turn can be used by the reasoner to deduce for example that the genre ‘Tennis’ is actually a sub-genre of “Racket sports” which in turn is a type of ‘sports’. Therefore, if the user’s request after the query refinement step contains the concept “Racket sports”, the reasoner will automatically include programs annotated with the genre ‘Tennis’, ‘Badminton’, ‘Squash’, etc. because they are all relevant for this particular query.

Secondly, the Database Management System uses WordNet to include synonym relations, which has shown to increase recall with relevant results [214] and definitely when rather general queries are used [246]. Further, note that we have chosen to only broaden nouns, which is a strategy also applied by others (like for example in [215]), limiting the number of keywords which are eligible for broadening. The keyword ‘car’ would thus be broadened with synonyms like ‘auto’, ‘automobile’, etc., but also with hyponyms like ‘limousine’, ‘taxi’, ‘jeep’, etc. Note that in our approach hyponyms have a lower priority. If many synonyms are found, hyponyms are omitted to constrain the number of keywords for which we need to search. However, words can have different contexts or meanings (polysemic words) which are modeled in WordNet by having different synsets. E.g., the word ‘bank’ can refer to a financial institution, but also to the slope immediately bordering a river. Therefore, including synonyms when searching for items depends greatly on the context of the word. Over the years, there has been extensive research dealing with this topic since it is relevant in all language interpreting applications. The involved research field is that of Word Sense Disambiguation, which already resulted in various techniques approaching the problem [134]. We however, have chosen a more practical and user-driven approach, by trying to learn from user
feedback how to improve our search strategy in the future. For example, initially the system does not disambiguate word senses when users search for content. In other words, all synonyms of different synsets are considered while searching for results. However, for every user-driven search we keep relevant search parameters (e.g. the original request, the query refinement results, how every result satisfied the restrictions, etc.) and more importantly, we also monitor how the user reacts on those retrieved results (e.g. which ones he clicked, watched, etc.). Then, after many different search iterations, we can extract from this feedback which synsets were considerably more often chosen by the user. Since we know with which keyword/synset every search result matched, next time we can conclude to only add synonyms from the previously most often chosen synset. For many keywords it is likely that users prefer one particular synset over the other(s), since many word senses are completely irrelevant for every particular domain. Eventually, user feedback, which is now used as input for the search strategy, leads to better results after time. I.e. the systems learns to know the domain and how users interact with it.

Further, all search specifics which are saved for search optimizations, can afterwards also be used to explain to the user why some results were included in the result set. By doing so, we support the user in the ability to scrutinize his results.

### 7.2.3 Content Filtering

The Content Filtering component is the last step in providing personalized results. This process further adapts the result set, of a particular search, by looking at the user model. In principle, this step is not so different from the query refinement process where we also make a selection, of the content to be retrieved, by means of a query. However, in the introduction we already argued that the combination of a query refinement step before data retrieval and a content selection step after data retrieval is favorable, both in terms of querying efficiency and filtering flexibility.

With this component we are able to remove or update items in the result set of a query, based on the user model. By combining a set of rules on one hand and one or more user models on the other, the result set can be further adapted to the current user(s). While most content filtering depends heavily on the user model, some filtering just improves the quality of the metadata by filtering redundant information (e.g. if two sources added a value for the same property, but one source is less knowledgeable). Cases in which content filtering is applied, include:

- **Removal of inappropriate results:** e.g. if the user’s age is lower than 16, then remove all items with the genres ‘Horror’, ‘Erotica’, etc.

- **Removal of metadata overlap:** e.g. if a program’s credits metadata was enriched by IMDb and the program was a movie, then we remove the original source’s information modeling the program’s credits. For example, for movie credits IMDb is the most knowledgeable source.

- **Addition/update of a program rank:** e.g. if a program has an extensive synopsis attached, the rank property is updated to push it higher up the ranking list (general rule) or if the program has a genre the user likes a lot, the program rises as well (user dependent rule).

- **Presentation related rules:** e.g. if a user uses his mobile, we can adapt the metadata to a small screen, by for example removing high resolution pictures or trailers from the metadata.

The Filters and Rules repository (shown centrally in Figure 7.1) provides the rules to filter items from the result set as response to a user query. Such a rule consists of an antecedent $P$ which describes a set of restrictions, and the consequent $Q$ describing what should be done with the result from $P$. The restrictions which are formulated in $P$ can refer to properties in the metadata (following the ontological schema), but also to properties of the user model. In this way we can for example express in $P$ that users younger than 16 years old should not be able to retrieve content which is annotated with genres like ‘Horror’ or ‘Erotica’. In this particular case, for the programs for which $P$ is true, the consequent $Q$ could be to remove those programs from the result set. The antecedent of the rule we model by means of a selection query which
retrieves all programs matching the constraints of $P$. The consequent is an action which includes for example ‘REMOVE’ and ‘UPDATE’. A rule is therefore defined like:

\[
\text{IF} \ <\text{program selection query}> \ \text{THEN} \ <\text{action}>
\]

In Figure 7.2 we see a SPARQL query which selects programs which are either annotated by the genres ifz:3.4.6.6 (Horror) or ifz:3.4.8 (Erotica), while the user is younger than sixteen years old (has a birthday which is more recent than the current time minus 16 years). This query is the rule’s antecedent which describes the path expressions in the ontological graph leading to the program properties describing the genres and the user’s property modeling the day of birth. Since in this case the consequent would be defined as ‘REMOVE’, the resulting programs of this query are deleted from the result set.

```
PREFIX ifz: <http://iFanzy.nl/Ontologies/TV-Anytime>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT { ?program }
WHERE { ?program rdf:type ifz:Program .
  ?program ifz:BasicDescription ?basicDesc .
  ?basicDesc ifz:Genre ?genre .
  FILTER(?genre = ifz:3.4.6.6 || ?genre = ifz:3.4.8)
  ?program ?x ?z .
  ?user ifz:hasCharacteristic ?char .
  ?char ifz:name ?name .
  FILTER(?name = "birthDay" && ?value > "1993-11-16T00:00:00"^^xsd:dateTime) }
```

Figure 7.2: Example of a SPARQL query finding inappropriate programs

However, removing content is just one of the different possible consequences of a rule. As introduced at the start of this subsection, a rule can also be used to update information. When items are retrieved after execution of a query, we rank them in an order taking for example the user model or the quality of the metadata into account. E.g. a higher value is given if the program has a synopsis of considerable length, if the program’s language or subtitles match the user’s native tongue, if the program has associated pictures and/or trailers, etc. The antecedent of these cases will be similar to the query in Figure 7.2. A query for example selects all programs where the spoken language (IFZ:Language) or the caption language (IFZ:CaptionLanguage) in the basic description of a program equals the user’s native tongue characteristic. The consequence is now, instead of ‘REMOVE’, an ‘UPDATE’ statement which is executed for every program which resulted from the antecedent query. This ‘UPDATE’ statement looks for example like “UPDATE increase(IFZ:Program.rank, 1)” which tells us that the program’s property ‘rank’ should be increased by ‘1’. The higher this value or rank eventually becomes, the higher we estimate its quality and relevance with respect to the user within this particular result set.

While the Query Refinement step conceptualizes and adapts the query to the user and the Database Management System broadens the search space through reasoning, the Content Filtering step filters and updates the result further improving the quality, and in some cases the appropriateness, of the result. We can say that after performing these three steps, the initial user query resulted in a personalized set of items, either matching the initial query or being highly related.
7.2.4 Recommender System

Previously, we saw that the average television viewer is a rather passive end-user. People want to enjoy their television experience and not being bothered too much with all kinds of settings and preferences, which is just more convenient on for example a Web platform. Therefore, we presented in Chapter 6 how we can build a user model, without requiring almost any user participation, which can help in personalizing the TV viewing experience. However, in the previous section (7.2.1), we learned that searching for content is a task which is actively driven by the user, i.e. the user wants to search for particular items by formulating a search request. Further, when search results are returned, the user needs to skim through this set to see which information satisfies him most. A more passive approach however could be that the system, when turned on, presents the user with all programs, currently available, which he most likely will appreciate in the current situation. Moreover, when these programs are found and served without any further user participation, it would greatly support the user, considering that the number of all available programs is just too big to go through, providing instant enjoyment.

Recommender systems assist and augment the natural social process to make choices without sufficient personal experience of the alternatives [201]. The ultimate goal of any recommendation system is to make sure that a user has no need to search himself anymore, as the systems exactly knows when and which items to propose. The only necessary input for a recommender system is some user information (depending on the approach), like for example the model discussed in Chapter 6. With such a user model, a recommender is able to predict to which degree a certain item in the data set will be appreciated by the user. As explained in [242], there are basically three categories of prediction strategies:

• Social-based prediction techniques: Analyze the behavior and characteristics of users without using knowledge about the data items. They use the known behavior and characteristics of the current user and other users to deduce the predicted interest of the item for the current user. Examples include among others Collaborative Filtering, Demographics Filtering, etc.

• Information-based prediction techniques: Sometimes also called content-based prediction techniques, analyze item \( I \) in comparison to other items and the knowledge about how the current user felt about those items to deduce the predicted user interest in \( I \). The most known example here is Case-Based Reasoning.

• Hybrid prediction techniques: A Hybrid prediction approach combines other known algorithms into one consolidated technique. Because every individual prediction technique has its own strengths and weaknesses, different techniques and algorithms can complement each other very effectively.

In our approach we use a hybrid prediction approach combining Demographics Filtering (DF) and Case-Based Reasoning (CBR), to which we refer by the name \( \text{Recc}_{DC} \). In this approach, the CBR component is dominant. It tries to predict the user’s interest based on previous behavior. However at cold start time, there is no previous behavior yet. Therefore, in Section 6.3 we already discussed some measures how to alleviate the cold start problem based on existing stereotypes and statistics. This technique therefore acts as the demographics filtering component in \( \text{Recc}_{DC} \). For more information about hybrid recommendation approaches, we like to refer to [57]. A possible and desired extension of this \( \text{Recc}_{DC} \) involves including the Collaborative Filtering technique. However, due to practical limitations which we discuss further in the text, it is not included yet. In the following paragraphs we discuss each of these three techniques more thoroughly.

Recommendation techniques

Demographics or stereotypes filtering [202] belongs to the group of social-based prediction techniques. A stereotype is basically a simplified conception of a group of people, who behave similarly in terms of key characteristics important for the target domain. Typically, these stereotypes are well understood within that domain. For the television domain, a particular stereotype must thus
be well understood in terms of demographics (like gender, age, education, etc.) but also concerning interests in program, genres, channels, etc. Demographics filtering is very useful for application areas in which quick but not necessarily completely accurate assessments of the user’s background knowledge are required. Stereotypes are used frequently to construct an initial first representation of a user profile [110]. Also in our approach, we mainly use demographics filtering at cold start time when user information is limited. In Section 6.3, we showed that we classify users in various stereotypes available through historical user statistics, IMDb ratings and MovieLens movie ratings. These stereotypes were constructed based on characteristics like age, gender and education.

In combination with demographics filtering we use Case-Based Reasoning or CBR. The principle of CBR, as described in [203], is simple: A case-based reasoner solves new problems by adapting solutions that were used to solve old problems. By mapping this to a recommender engine it reads: “trying to predict how a user feels about a new item, by looking at how items, rated in the past, relate to this new item”. CBR basically approaches the problem from the item’s perspective rather than from the user’s. CBR is, according to [242], especially good in predicting how interested a user is in the same types of information or slightly different versions of the same information. In the TV domain we are often confronted with this particular situation, different episodes of the same series for example only differ slightly in their metadata descriptions. Therefore, the key aspect of CBR is determining the similarities and differences between two items, which’s calculation depends a lot on the structure of the domain.

Another well-know and widely used recommendation algorithm is Collaborative Filtering (CF) [200, 122]. Collaborative filtering, is based on the cooperative subjective evaluations of items by other people in a group. The algorithm assumes that if a user likes a number of items just as much as a group of other people, the user probably has the same taste as this group. Knowing this, we can assume that when predicting the interest of this user in a yet unrated item, this group can again be a good predictor. In contrast to the CBR technique, which approaches the problem from the item’s perspective, CF solely looks at the user’s perspective. Therefore, in collaborative filtering, they do not compare items but rather the users themselves. CF has a number of advantages over content-based approaches. CF is an accurate domain-independent prediction technique, especially for content that cannot easily and adequately be described by metadata (like e.g. TV programs), it has the ability to filter items based on quality and taste, and it can provide serendipitous (i.e. surprising) recommendations [242, 166, 121]. CF however, also suffers from three disadvantages, namely the Sparsity problem (most users do not rate most items, making it hard to find enough similar ratings), First-rater problem (an item cannot be recommended unless it has been rated before) and the Gray Sheep problem (in small or even medium user communities there are always individuals who do not really fit in any of the groups and therefore almost never get good recommendations [63]). However, the gray sheep problem does not affect CF only, also demographic filtering suffers from it. Moreover, in the TV domain the problem is even more pronounced. A television recommendation engine is somewhat special in comparison to general recommender systems. Most recommendation systems commercially deployed like e.g. Netflix\(^3\), Amazon\(^4\), Last.fm\(^5\), etc. work on top of a relatively stable item set. The majority of the items in the set stay there for a long time, only sporadically items are removed and, in comparison to the already present set, relatively few new items appear over time. In contrast, in the television domain, apart from possible Video-On-Demand (VOD) sources, programs constantly come and go. A user can not just decide to watch a program again which was broadcast yesterday, and every day several hundreds of new programs appear (depends on the number of channels maintained) while users only have a few days to actually rate them. According to [123], most CF algorithms have been designed specifically for data sets where there are many more users than items. Therefore, such algorithms may be entirely inappropriate in a domain where there are many more items than users. As a consequence, to perform CF, a relatively large group of active users is required, which is currently lacking in iFanzy. Therefore, iFanzy does not deploy CF yet, until some of the previous requirements are met. Already many others have attempted to

---

\(^3\)http://www.netflix.com/
\(^4\)http://www.amazon.com/
\(^5\)http://www.last.fm/
make a recommendation engine for movies, e.g. in systems like FilmTrust [106], MovieLens[122], Recommendz [102], etc. However, much less have tried including television programs in general.

**Recommendation employment**

Looking back to the Adaptation Loop presented in Figure 7.1, we see the Recommender component defined as a part of the Content Filtering component. The query refinement process (as described in the previous section), is no different when a search action is performed or when a recommendation is requested. After all, just like when searching for specific items, users are able to place restrictions on a recommendation as well, to get for example a recommendation for a well-defined subset of items. A user can for example ask a program recommendation for tomorrow or only considering sport programs or only programs from a particular VOD source or a recommendation of programs in which “Brad Pitt” plays, etc. Parsing such a restricted recommendation therefore follows the exact same steps as explained in Section 7.2.1.

When the recommendation query is executed, results are sent to the Content Filtering component. Here the Recommender component, employing ReccDC, calculates a recommendation score $R_S$ for every item $I$ in the result set. In ReccDC, every item is evaluated against the user’s model, with current context being taken into account. If the user rated $I$ before, this value is returned as $R_S$. However, if the user did not rate $I$ before, we need to approximate the user’s interest by means of user likings. Previously in Section 6.2.3, we saw that we calculate likings for every individual $A$ referenced by an object property of $I$. We also saw that one such individual could get several likings depending on the source of the liking. After all, different events led to different likings (see Formula 6.2.1 and 6.2.2), data mining algorithms could generate likings (see Formula 6.2.3), etc. Furthermore, whenever regular likings are missing, the ReccDC algorithm asks the demographics filtering (DF) component to fill in the gaps. This component returns a stereotype liking (see Definition 7.2.1) which is a consolidated value generated for the stereotype, to which the user belongs, through the different cold start approaches presented in Section 6.3.

**Definition 7.2.1. Stereotype Liking**

A stereotype liking $SL(u, r, c)$ represents the degree of potential interest of a user $u$ in resource $r$ under context $c$, as determined by the stereotype group to which $u$ belongs.

Consequently, for every individual $A$ referenced by an object property of $I$, we define the general liking $L_G$ expressing the generally predicted user’s interest in $A$. Formula 7.2.1 shows the calculation of $L_G$.

**Formula 7.2.1. General liking**

Let $r$ be a resource, $L = l_1, \ldots, l_n$ be the set of likings of user $u$ previously generated for $r$ in context $c$, $W = w_1, \ldots, w_n$ be the set of weights associated to every liking source and $SL(u, r, c)$ the stereotype liking. Then the general liking $L_G(u, r, c)$ becomes:

$$L_G(u, r, c) = \begin{cases} 
\frac{\sum_{l_i \in L} w_i \cdot l_i}{\sum_{i=1}^{n} w_i} & \text{IF } L \neq \emptyset \\
SL(u, r, c) & \text{Otherwise}
\end{cases}$$

Having defined the general liking $L_G$, we can now formulate the calculation of the recommendation score $R_S$. Firstly, if the user rated $I$ before, this rating directly determines the $R_S$. If the user did not rate $I$ before, we investigate all the item’s object properties. For every individual $A$ referenced by an object property of $I$, we retrieve the general liking (calculated following Formula 7.2.1) which, together with an associated weight, then contributes in the approximation. If for some reason not a single individual has an associated $L_G$, we try to find a stereotype liking $SL$ (as defined in Definition 7.2.1) for item $I$. Lastly, if also no stereotype liking can be found, we return ‘unknown’. In Formula 7.2.2, we see the algorithm to calculate $R_S$. 
7.2. A PERSONALIZED DATA ACCESS STRATEGY

Formula 7.2.2. Recommendation score
Let \( r \) be a resource, \( A = a_1, \ldots, a_n \) be the set of individuals which are connected to \( r \) by object properties, \( W = w_1, \ldots, w_n \) be the set of weights associated to each of those object properties, \( v_r \) the rating of user \( u \) on \( r \) in context \( c \) and \( SL(u, r, c) \) the stereotype liking. Then the recommendation score \( RS(u, r, c) \) becomes:

\[
RS(u, r, c) = \begin{cases} 
  v_r & \text{IF } u \text{ rated } r \text{ in context } c \\
  \frac{\sum_{i=1}^{n} w_i \cdot LG(u, a_i, c)}{\sum_{i=1}^{n} w_i} & \text{ELSE IF } L \neq \emptyset \\
  SL(u, r, c) & \text{ELSE IF } SL(u, r, c) \text{ exists} \\
  \text{unknown} & \text{Otherwise}
\end{cases}
\]

In normal execution of this algorithm, every attribute \( A \) has an associated weight which determines how heavy it contributes to the final value. These weights are carefully determined by domain experts in combination with previous user test results. Further, optionally we can set these weights to be determined dynamically by an attribute selection algorithm like e.g. the Correlation-based Feature Subset algorithm [112]. Such algorithms are known in the data mining field to calculate the predictive power of a particular attribute. This knowledge allows for example to remove redundant attributes which could potentially confuse the learner. The output of an attribute selection algorithm is a table showing all attributes together with an indication of their predictive power in the current data set. This information can therefore be used to influence the weights, which then slowly evolve favoring the attributes which are the best predictors for this particular user. By periodically running this algorithm over all the user models in the system, we see a clear indication of which attributes are the most predictive. In iFanzy, and therefore the television domain, we could see that one of the most predictive attributes was the ‘channel’ attribute showing where the program was broadcast. This did not come as a surprise since television channel managers deliberately focus their channels (and thus programs running on those channels) to particular user groups. E.g. channels focusing on young males show football and action movies, while channels focusing on mature women broadcast soaps, cooking shows and wellness and lifestyle programs.

Our ReccDC algorithm, which combines Demographics filtering and Case-Based Reasoning, has shown to deliver good results (in Chapter 10 we show some evaluation results). However, there are also some concrete limitations. Firstly, this technique depends heavily on the quality and completeness of the available program metadata. Although we try to enrich this data as much as possible, it can occur that a program remains poorly annotated. In such a case the ReccDC algorithm has difficulties finding similar programs and eventually ends up with low support and confidence to calculate a recommendation score. Secondly, in principle there is a considerable performance penalty with CBR systems, because every item needs to be compared to the user model (or set of user models in case of a group recommendation). However, there is also a lot of room for optimizations. For example, in every user model all the likings are precalculated so that when they are needed, they are ready to be used. However, when a user performs an action triggering an event in the user model, like for example when the user rates a program, all relevant likings should be updated. E.g. rating an action program, requires an update of the liking for the genre ‘Action’. Therefore, after every user generated event, all related likings are flagged. Later, in times of lower server load, like for example at nighttime, we recalculate these flagged likings. Further, all demographical statistics and stereotypical information is cached in memory such that it can be accessed quickly if necessary. The items in the result set which need to be compared to the user model on the other hand, also reside in memory enabling a fast comparison between the item’s metadata and user model information.
7.3 Conclusion

In this chapter we have presented our approach towards context-sensitive user-adapted data retrieval, which we illustrated with our knowledge of the television domain. Personalization fitted in our approach, presented in Chapter 4, to show that given a rich data model, integrating data from various heterogeneous sources, in combination with an extensive user model, can lead to a structural domain-independent approach allowing the user to access this data in a personalized way.

In this chapter, we explained a strategy to facilitate personalized data retrieval starting from a simple user request until the final result is returned. This strategy (outlined in Figure 7.1) included the refinement of the initial user request, a query handler including a semantic reasoner and a content filtering step, together facilitating this data retrieval. To do so, all three steps make use of three data repositories serving user model data, ontological knowledge and various filters and rules, in order to make the result as personal as possible. While a user obtains data either via searching or asking for a recommendation, in both situations we followed the same rigorous principles and strategies to personalize the results, including as much user knowledge as possible. However, the recommender differs in that perspective, that instead of ranking results based on for example the quality of their metadata, results are ranked depending on a prediction indicating how much a user will like every item.

Further, every step of this process can be applied to a group of users as well. This is necessary when for example a family watches television together, and the system should return a set of programs maximizing the group’s satisfaction. However, the difficulty here lies in how several user models can be combined most efficiently, trying to keep the group’s satisfaction as high as possible. Currently we use a simple averaging strategy, where ratings of the users for the same concepts are averaged. Such grouped user models can be kept in the system indefinitely if that particular group joins up often, to watch television together. Moreover, the group models can even get updated automatically every time the profile of one of the members of the group changes. For more strategies creating group user models, we would like to refer to [160].

In general we can say that this chapter provides an extensive overview of an approach, able to provide context-sensitive personalized search and recommendations, given an integrated item set harboring well-described items (as described in Chapter 5) and an extensive user model (as described in Chapter 6). Therefore, this approach answers research question 6 (How can we provide user-adapted data access given a well-defined domain model and a comprehensive user model?).

7.4 Related Work

In this chapter, we focussed on how we can personalize the user’s quest for information. Therefore, in this related work section we zoom in on similar adaptation and personalization frameworks and methodologies exploiting well-structured and preferably semantic-rich data. However, personalization is a very broad research field including areas like information access, filtering, recommendation, search, navigation, etc. Nonetheless, in this overview we first try to list a general overview of personalization, followed by more specific research in the field of personalized data access.

7.4.1 The broad field of personalization

The concept of ‘adaptation’, “user-adaptive systems” or “adaptive interfaces” has been around for quite some time now. In [209], Schneider-Hufschmidt et al. present a state-of-the-art report and taxonomy for the field of adaptive interfaces together with discussions, summaries and evaluations of prototypes and systems. This paper shows the results of a workshop organized to develop a coherent view of the results accomplished in this field, bringing together a number of well-known researchers in the area of adaptive user interfaces. Some years later, this field of adapted systems contributed to the introduction of Adaptive Hypermedia systems where hypertext documents were adapted to the user. In [53], Brusilovsky provides an overview of existing work on adaptive
hypermedia, classifies them based on different methods and techniques, and describes the most important ones more thoroughly. Brusilovsky further describes three *stages* in a general adaptation process encompassing: collecting data about the user, processing the data to build or update the user model, and applying the user model to provide the adaptation. Similarly, our personalization strategies in combination with the proposed user modeling approach, follows this pattern. In the same period, also the Adaptive Hypermedia Architecture (AHA!) was conceived [71]. AHA! was initially developed to serve as an educational system at the Eindhoven university of Technology (TU/e), but was soon turned into a *general-purpose* tool aiming at bringing adaptivity to a wide variety of applications such as online information systems, museums, shopping Web sites, etc. [224]. AHA! provides two kinds of adaptation: *adaptive presentation* (inserting/removing fragments depending on the user model) and *adaptive navigation support* (link adaptation depending on the domain model combined with the user model).

However, while adaptive hypermedia systems were dealing with relatively simple hypertext documents interconnected through links, a new dimension opened up with the introduction of Semantic Web [80]. Obviously, the connection between user models and semantically enriched data was a logical step. One of the pioneering projects, exploiting semantically annotated information in combination with user modeling, is the Hera project [243], a methodology to support the design and engineering of Web information systems. Initially Hera only adapted the presentation generation step where they deployed conditional inclusion of fragments (slices in their context) and link hiding. Later Hera reincarnated in Hera-S with an even more pronounced focus on personalization, RDF querying and storage [128]. For example, their application model allows dynamic personalization, such that for example navigational access to the data can be personalized to a user and adapted for a specified context. In [146], Kobsa et al. present a comprehensive overview of techniques for personalized hypermedia presentation, describing the data about the computer user, the computer usage and the physical environment that can be taken into account when catering hypermedia pages to the needs of the current user.

As the concept ‘personalization’ became more popular and in most cases more essential, it was picked up in many different domains. *E-learning* for example is the field of technology enhanced learning where teachers and/or students are supported in their educational tasks. Already many years of research on adaptive E-Learning has led to encouraging results [54]. However later on, E-learning was bridged with adaptive hypermedia techniques [55], improving the results and presentation, and continues being developed nowadays using state-of-the-art techniques like for example community-based recommendation systems [93]. Further, others combine E-learning systems with semantically annotated data. In [79], Dolog et al. show how to realize personalized learning support in distributed learning environments based on Semantic Web technologies, trying to fill the gap between current adaptive educational systems, with well-established personalization functionality, and open dynamic learning repository networks. They propose a service-based architecture to establish personalized e-Learning including a *Query Rewriting Service* (adding additional constraints based on the user profile), a *Recommendation Service* (to annotate learning resources according to the educational state of the user) and a *Link Generation Service* (connecting a learning resource to other resources, or to a context). [14] continues this line of work with a specific emphasis on the interoperability of such systems.

Besides E-learning, also in other domains researchers try to personalize their functionality. In the E-culture domain for example, it is all about bringing art and culture closer to people by filtering the pieces they will appreciate. In [204], Roes et al. present their personalized museum guide, developed in the CHIP project. With their system they provide an intuitive bridge between their online Web site, offering personalized interaction and art recommendations, and on-site experiences, presenting a personalized tour through the museum. In [159], Martin et al. describe SCALEX, a personalized multimedia information system for museums and exhibitions. SCALEX offers a museum toolbox, enabling the combination of digital content, with real exhibition objects. The presentation of the digital media is directly coupled to the interests of the visitors, to realize personalized and adaptive knowledge based exhibitions. In [44], Bowen et al. advocate to use personalization in museums to alleviate their traditional communication paradigms (where one group is present and the other absent), and let users re-find the joy of visiting the museum.
Further, we see strategies towards personalized news services by using the combination of adaptive hypermedia techniques with the ideas of loose and strict ontologies [66]. Personalization provided by recommending items to a specific person, has already been applied in a variety of domains like E-commerce, news, television programs, music, books, movies, etc. by many commercial companies like Amazon, Last.fm, eBay, Netflix, Outbrain, etc. In [207], [10] and [169] a nice overview is provided of recommendation techniques for personalization in the E-commerce, television program guides and usenet field respectively. In [199] on the other hand, Reinecke et al. show how you can adapt for example a user interface based on the cultural background of a person. For further information about personalization for the Semantic Web, we like to refer to [20].

From a different perspective, we also want to refer to some points of caution when performing user modeling to support personalization. In [249], Wang et al. discuss the issue of privacy and show how these constraints may affect Web-based personalized systems. They believe that every personalized system needs to take users’ privacy concerns into account, as well as privacy laws and industry self-regulation that may be in effect. As solution they present a dynamic privacy-enhancing user modeling framework as a superior alternative, which is based on a software product line architecture. In [59], Carmichael et al. discuss personalization and user modeling in ubiquitous computing environments, however with a strong emphasis on scrutatable modeling. They see scrutability as a necessary foundation for user control over personalization. Allowing the user to examine and understand his own user model, will involve the user in the process and consider the application more trustworthy.

Lastly, a more general domain-independent approach towards context-aware, ubiquitous and pervasive applications was presented in [17]. There, Assad et al. present the PersonisAD framework which relies on its defining foundation: a consistent mechanism for scrutatable modeling of people, sensors, devices and places. This framework facilitates the quick creation of new context-aware applications featuring distributed models with active elements, which can get triggered when relevant events occur. They applied their strategy in two context-aware applications, including MusicMix which plays music based on the preferences of the people in the room and MyPlace, which informs people of relevant details of the current environment. The idea is to measure the context of an environment by means of sensors and to react properly in a predefined way. Via their accretion/resolution mechanism, which associates triggers or rules with components, new evidence can be added to any component whenever a rule succeeds, which in turn can trigger other rules or effect the environment. Such a system could be useful in the TV environment as well. Rules could for example automatically change the TV program, subtitle language or screen brightness whenever a new person enters the environment.

7.4.2 Personalizing data access

As discussed in the previous subsection, personalization is a very broad field, relevant in many different domains. In this chapter however, our personalization approach focussed primarily on personalizing data access given the data model and user model described in previous chapters. Therefore, in this subsection we take a closer look at related work specifically focussing on this narrow part of the personalization spectrum.

In [105], a personalized searching approach is presented which is very similar to ours. In their approach, they advocate two different methods leading to personalized search: re-ranking (by classifying each of the search results into the different categories of a reference ontology, and comparing the user profile’s values for the categories selected as being the most similar to the document) and filtering (excluding those documents whose revised ranking values fell below a threshold). Also in other research like [228], re-ranking based on the user’s profile is used. In [228] however, the user profile is built based on previously issued search queries and visited Web pages. Also in [190], an approach is taken similar to our own. With Outride, a personalized search system, the personalization engine sits between a user interface and a search engine. Upon a user query, first the query is extended: e.g. if a user is looking at a series of pages on car information and searches for ‘contour’ the system may augment the query by adding the term ‘car’ or ‘ford’ to
provide the user with results about the Ford Contour car. Based on the user model, the system can reinforce the query with similar terms, or suggest results from prior searches. Once the search engine has processed the query, the results can be individualized. Information can be filtered or re-ranked based upon information in the user’s model and/or context.

In [225], Sugiyama et al. try to personalize search results using a user profile built by monitoring the user’s browsing history. Every time the user browses search results, which were returned after a query, the user profile is updated either purely based on browsing history or on modified collaborative filtering. This continuous process leads to a user profile usable to personalize the results of the next search action. In [154] however, Liu et al. propose a method to map a user query to a set of categories, which represent the user’s search intention. A user profile and a general profile are learned from the user’s search history and a category hierarchy respectively. Based on these two profiles the user’s search is personalized by deducing the appropriate categories for this search and this user.

A rather different approach, based on the PageRank algorithm, is proposed in [116]. While the original PageRank algorithm computes a single PageRank vector, using the link structure of the Web, to capture the relative importance of Web pages, [116] proposes a topic-sensitive extension. By computing a set of PageRank vectors, biased using a set of representative topics, they capture more accurately the notion of importance with respect to a particular topic. Then, they determine the topics most closely associated with the user query, and use the appropriate topic-sensitive PageRank vectors for ranking the documents satisfying the query.
Chapter 8

Data Retrieval Performance

In the beginning, the Internet was a pretty simple and straightforward concept. A user could interact by entering a URL, which in turn delivered a Web page containing a piece of text, including some limited markup and links to other pages. The system was pretty straightforward and could scale up relatively easy. Nowadays, after fifteen years of explosive growth, the same principle of the Internet still holds, however, the operational complexity has grown at least proportionally and is predicted to rise even faster [256]. Currently, a Web site is an interwoven complex depending on various external services and data sources, constantly updating its content to provide a realtime overview. Many Web sites show, by means of little “window gadgets” or widgets, for example a live weather forecast for the region, live traffic jam information, arrivals of planes on the airport, the status of specific stock markets, many live and changing advertisements, connections to social networks like Twitter, Facebook, etc., links to shops and prices of related objects and much more.

Besides the fact that Web sites are growing more complex, thereby contributing to the increase of Internet data traffic, they are also growing in number. At the end of 2009, the December survey of Netcraft\(^1\) already showed 233,848,493 active Web sites, growing steeper than linear. On top of that, also more and more people start using the Internet with at the end of 2009 already 1,802,330,457 active Internet users\(^2\) (which is the fivefold of the number in 2000). Lastly, the size of all data available on the Internet is probably growing the fastest of all, albeit difficult to measure. In 2005, Eric Schmidt (Google CEO) said that, from the data recorded by the search engine, it seems that, at this specific moment, the Internet is made up of about 5 million terabyte\(^3\). However, we can safely assume that this number nowadays will be a factor higher, without any indication of leveling off any time soon. Nonetheless, this growing amount of data available on the Web or at least available in the application domain, is probably the biggest worry of any application and reconfirms the importance of a good data management system.

To cope with this evolution, systems and services are required to be set up in a scalable and well-performing fashion. A system’s scalability connotes the ability of the system to accommodate an increasing number of elements or objects, to process growing volumes of work gracefully, and/or to be susceptible to enlargement [43]. The system’s performance is indicated by the amount of work that can be done within the limits of a set amount of time and resources. For example, to speed up browsing, Ajax was introduced as a technique to reload a small subpart of a page asynchronously, avoiding a full page reload. Considering our belief that personalization of services, information access, data filtering, navigation, presentation, etc. will become more and more prevalent, and considering the additional data and computational complexity to provide that personalization, we expect the scalability and performance of a system to become even more crucial in the future, ensuring the system’s long-term success.

In this chapter we look at a number of optimization techniques which increase the efficiency when querying the database. Having a large metadata database, obtained after the integration of

---

\(^2\)http://www.internetworldstats.com/stats.htm
various sources, in combination with potentially complex queries, generated by the personalization process, can sometimes lead to long querying times. Optimizing both the database structure and querying approach, decreases querying times tremendously. The provided optimization strategies address research question 7 (Given our approach towards user-adapted data retrieval, how can we optimize data retrieval efficiency in terms of querying speed?).

### 8.1 Introduction

Our approach, described in Chapter 4, encompasses many heavy-duty tasks. Some are independent of user activity (like for example fetching metadata, integrating information from different sources, mining user model patterns, etc.), and can therefore be scheduled during periods of low user activity. Other heavy-duty tasks however, which are directly triggered by the user on-the-fly, have a more conspicuous influence on the system’s overall performance. Tasks like searching for content or obtaining a recommendation encompass a number of different user-dependent steps, like outlined in Chapter 7, which are slightly more complex to perform due to the inclusion of user information. However, most of the steps making the data retrieval process personal, like e.g. personalizing the query and filtering the results afterwards, have a limited impact on the system performance. This is largely due to the fact that necessary resources for these tasks (e.g. ontologies, a user model, etc.) are very controllable resources. Ontologies for example are very manageable in size, but also the speed in which they grow is very limited. This makes them ideal to maintain in fast “in-memory” repositories. A user model is more susceptible for updates and evolution. However, as we saw in Chapter 6, all user model likings and patterns discovered are materialized in the user model, such that they can be queried easily and without further calculation.

The step in the personal data retrieval process where most optimization can be obtained, is when querying the integrated metadata database, described in Chapter 5. Taking the amount of metadata and ontological knowledge retrieved from different sources, this database can potentially grow to a considerable size. In Section 8.2, we explain how different optimization techniques influence the performance of the database(s). To show the improvement, we start with a naive database implementation and progressively grow towards the end-result as gradually different optimization techniques are applied. These optimization techniques are general and improve the performance in any domain following the approach proposed in Chapter 4. To illustrate query performance, we use some representative and often occurring queries in the television domain.

### 8.2 Query Optimization and Performance

Many Web applications today are characterized by the integrated use of data from several already available sources or applications. Their engineering must therefore also address the efficient data integration and querying paradigm. To this end, more and more of them use, just like us, both Semantic Web techniques to model and Semantic Web data sources to enrich their own legacy data. When we observe current Semantic Web implementations, we see that most of the applications using Semantic Web data do this on a relatively small scale. They for example try to show a smart reasoning technique, integration with a new source or the application in a new domain, in which a large data set is not necessary to prove the point. Slowly, more and more large-scale Semantic Web-based applications are being developed, harboring large amounts of semantically enriched data. This is partially sparked by the fact that more and more large data sources are becoming available, either natively in RDF or after a transformation of the original source (e.g. Wikipedia), for integration into the Semantic Web. However, all of these applications have to face performance problems being introduced by such large data sets (e.g. in the Billion Triple Challenge, which is a recent part of the regular Semantic Web Challenge, they target exactly this problem). In Chapter 5 we already discussed some of these large sources which we utilize, like DBpedia and IMDb which have over 274 million and 66 million triples respectively. Including such data sources into a Web application therefore requires a well-considered database and querying approach.
8.2. QUERY OPTIMIZATION AND PERFORMANCE

A straightforward but naive first engineering approach implementing such Web applications, is to put all RDF data in one big repository. Having such a single repository, makes that queries do not need to be split or distributed and can be sent directly to the database, which will deliver the final result. In this section we first discuss the performance of such a repository, as point of reference. Afterwards we discuss different approaches trying to improve it. We show for example how well-known database techniques like vertical and horizontal decompositions improve query times in the RDF data set. Further, we show the significant and sometimes spectacular impact of using tools and technologies like relational databases, for well-structured parts in the data set, keyword indices to improve keyword search, etc.

8.2.1 A naive first approach

As we saw earlier, real-life data sources can be huge. Consequently, combining several of these sources, thereby aligning data to be able to infer additional knowledge, potentially induces performance considerations. During our experiments we noticed that in this setting, using realistic queries and data sources, performance indeed became a critical issues. We also saw, for example when discussing with tool and technology providers, that these performance problems in Semantic Web-based applications did not yet get the necessary attention to solve them in general. This might be in part because many such applications are created in a research setting, where examples are used that avoid the typical performance problems that arise when employing large RDF data sets. The straightforward and most common approach, which is successfully applied in such smaller-scaled projects, is to combine all data, including instances and schema(s), in one single data source. However, applying this approach in the television domain with the iFanzy use case, leads to a data set of over 400 million triples. For reasons of verification and comparison we tested this very naive approach where all our data was put into one big repository, even including data sets like WordNet and GeoNames. Although we expect results to be bad, we can still use them to act as a baseline performance level to which we can compare results after various optimization steps are performed. To start this performance benchmark, we constructed four typical information retrieval queries, for the television domain, with increasing complexity. These queries will be used throughout this section to evaluate and compare different optimization steps. The outline of these four queries is defined as follows:

- **Q1**: The first query retrieves all metadata of program $P$ with a maximum path expression depth of five. E.g. get all metadata for the program "The Godfather" five levels deep in the data graph. Due to inferencing, all other metadata referenced from $P$ via the $\text{OWL:sameAs}$ relation like for example from DBpedia, IMDb, etc. (as explained in Chapter 5) is included.

- **Q2**: The second query is a search query retrieving all programs with restrictions on the genre and geographical locations. E.g. get all programs with the genre `Action` and produced in `USA`. Due to the transitive property inferencing also programs are included which are annotated with a genre narrower than `Action` (e.g. “Sci-fi” or `Thriller`) and a geographical location narrower than `USA` (e.g. `California` or “New York”).

- **Q3**: The third query is a search query retrieving programs with restrictions on the genre and geographical locations, and an extra restriction for the keyword $K$ in the title. This query is basically the same as $Q_2$, however there should also be a positive match of the keyword $K$ in the title of the program.

- **Q4**: The fourth and last query is a search query retrieving programs with restrictions on the genre and geographical locations, and an extra restriction for the keyword $K$ in all literal properties with a maximum path expression depth of five. This query is basically the same as $Q_3$, however the keyword $K$ can now match with any literal property (maximally five levels deep) connected to that program (e.g. actor names, synopsis, keywords, etc.), instead of only with the title.
From the description of these queries, we can see that with every step an additional level of complexity is added. As for all experiments, each query was executed several times (query result caching disabled) and the average execution times are reported. The resulting times for these four queries are shown in Table 8.1. Although combining the different data sets was a necessary and beneficial step from a point of view of increasing the available knowledge, as we can see from the results applications employing huge semantic data sets suffer from considerable performance problems. Even for the simplest queries, execution times quickly exceed the acceptable thresholds for realtime environments. All queries ran on a Linux machine with 1GB ram and an Intel(R) Pentium(R) 4 CPU (3.0GHz) running Sesame as an RDF storage and querying framework, OWLIM as reasoning engine and Java as the host programming language.

### 8.2.2 Decomposition in Sources and Querying

After having obtained the baseline performance of the naive approach of putting everything in one large repository for reasons of comparison, a first approach to increase the database performance is to split up (distribute) the main data set into several smaller sets. By splitting off strategically chosen parts of the database, we can make the whole more manageable, obtaining several smaller RDF repositories. The main method of operation is to consider which smaller parts of the data are queried regularly together: splitting them off in a smaller store achieving an increased performance for those queries, while maintaining the possibility to link and combine query results at the global level, can lead to an increase of performance for the overall system. Drawing from work in databases (e.g. [198], p. 653 and 674–675), such decompositions can be performed in two ways. Vertical, property-based decomposition in databases is based on the schema; instances related to certain classes and properties are split off from the data set. Horizontal, instance-based decomposition is based on the resources: a set of instances with the same schematic structure are divided in \( x \) parts and split off from the main data set.

#### Vertical Decomposition

Vertical decomposition is based on the schema. Certain classes and properties, together with their instances, are carefully chosen and split off from the data set. We applied this idea to split the data source, grouping data that conceptually belongs together. This grouping was based on the expected queries and how they use the original sources. For example, iFanzy uses WordNet specifically to obtain synonym data. Therefore, it is logical to keep WordNet separate, which means that only one (smaller) data set has to be queried when this synonym information is needed. Next to WordNet, also some enrichments sources like DBpedia and IMDb were split off to separate repositories. Obviously, splitting off data sets has consequences for the queries: they need to be (transparently) split up, fired to the partial data sets, and their results combined (in case of DBpedia and IMDb, which are referenced via OWL:sameAs properties, the query does not need to be split. Rather, the data management logic detects these properties and in turn triggers queries to those sources based on the corresponding URIs). Because vertical decomposition possibly distributes several properties of a class over different data sets, the initial query has to be split up along the same distribution. This entails identifying

<table>
<thead>
<tr>
<th>Query</th>
<th>Average execution time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>31.526</td>
</tr>
<tr>
<td>Q2</td>
<td>138.354</td>
</tr>
<tr>
<td>Q3</td>
<td>224.663</td>
</tr>
<tr>
<td>Q4</td>
<td>+1.200.000</td>
</tr>
</tbody>
</table>

Table 8.1: Average execution times of typical iFanzy queries
which properties reside in which data set, and subsequently isolating these properties (and the conditions acting upon these properties) in a single (partial) query. To combine the results from such partial queries, the relation between these queries has to be considered. Path expressions referring to a common subject or object in a triple (i.e. specified by a shared variable in the partial queries) give rise to partial queries which depend on each other: the results of one partial query influence the result of the other partial query. In Figure 8.1, we see an example (here in the Sesame RDF Query Language or SeRQL) of how such queries can be decomposed. On the left hand side we see a path expression of length two, which is distributed over two sources by means of shared variable \( o_1 \). On the right hand side, we see three path expressions of length one which are all sharing the same triple subject \( s_1 \). However, after execution of the resulting partial queries, the individual results need to be combined to obtain the final result. The most straightforward way is by performing a join on the result sets, using the shared variable(s) as the join attribute(s). These join attributes are included in the SELECT clause of every partial query so that they can be compared afterwards using the join condition. In Figure 8.1, we see that variable \( o_1 \) and \( s_1 \) respectively are included in every partial query.

However, in some cases smarter and more efficient strategies can be employed to combine the result sets. For example, if it is known beforehand that one of the result sets from a partial query will be significantly smaller, it is more efficient to include the results of this set directly into the query to the other data set, thus additionally constraining it and significantly reducing the execution time of this query. This approach gives rise to a query pipeline (and thus a sequential process), where the results from the previous step are used as input for the next step. Take for example a query asking for all programs which have the keyword \( K \) in their title. Due to the query refinement steps, originally the query sent to the database will ask for all programs which have \( K \) or any of \( K \)'s synonyms in their title. However, by having the WordNet set in a separate database, we can first ask for all synonyms of \( K \) and use these results as input in the program fetching query. There are basically two ways in which this can be concretely implemented: either by including all of the results (\( K + \) its synonyms) in the query at once (in the WHERE clause), or insert each of the results separately and thus execute the query multiple times. We have conducted experiments on this using a slimmed down version of the IMDb data set. In these experiments, we manually included synonyms from the WordNet data set in the queries to the IMDb data set, experimenting with both aforementioned techniques. The results can be found in Table 8.2. In the first approach, all of the synonyms are put in the query at once, and the query is subsequently executed. In the second approach, the query contains only one synonym at a time and is executed for each synonym; consequently, the average execution time of the second approach represents the combined execution times.

Figure 8.1: Possible query decomposition (SeRQL)
As can be seen from the results, the first approach can be favored over the second. This is because for each movie resource in the data set, the RDF graph has to be traversed to make the necessary keyword comparisons. When comparing these keywords all at once for each resource (as is the case in the first approach), this traversal only has to be done once, while in the second approach all the resources are checked (and therefore the graph traversed) for every query. However, this approach can potentially lead to a query with a large WHERE clause (due to all the synonyms included there). Therefore, we also tested how the query engine in general keeps up with a growing number of boolean expressions in the WHERE clause, and its impact on the query performance. Here we found out that there is no impact at all up to 100 disjunctions. Moreover, looking at some WordNet statistics, we found that every word on average has 1.76 synonyms and the maximum number of synonyms currently held by one word is 28.

### Horizontal Decomposition

As was already mentioned before, horizontal decomposition decomposes a data set in two or more groups, identified by specific properties within the set. Imagine having a large database containing all the people in this world, a decomposition could be to make a separate database per country (using the nationality property). Or, to get a more even dispersion, we could also split the database based on the “day of birth” property forming a new decomposition per decade.

In our case, we observed for example that querying the IMDb set presented a major performance bottleneck and was too slow for real-time query answering. This is not really surprising considering the size and complexity of the set. From the other side however, links, by means of the OWL:sameAs property, from regular programs to IMDb instances occur frequently. Therefore, IMDb information needs to be queried every time program metadata of one of those programs is required. However, by studying the IMDb queries actually performed by customer usage, it became obvious that only a relatively small subset of movies was frequently requested while the majority was not. People in general mostly tend to search for popular blockbusters. This knowledge gave rise to a preference for a horizontal decomposition rather than a vertical one. Moreover, decomposing the IMDb repository vertically (e.g. by splitting off some properties) still implies that we need to search through all movies to obtain the metadata of one. Therefore, we chose to perform a horizontal decomposition based on the popularity of the movies. In accordance with the desired application functionality, we used the votes that were issued by the IMDb users as criterion for the popularity of movies. Hence, people mostly vote on the known movies, good and bad. Based on a stepwise restriction on the number of movies via their popularity (see Table 8.3, column 1 and 2), we compared the number of user queries still answerable by this subset with the average query response time for a typical query to the IMDb data set (see Table 8.3, column 3), and decided to split off movies that have more than 500 votes from the main IMDb data set and put them in a separate data set. This new set contains around 11,500 movies which can be queried swiftly. However, since this new trimmed subset only contains a fraction of the total number of movies, this data set is not always sufficient to answer each user query. In that case, the main IMDb data set, which we still keep available, is consulted. Evidently, every time we need to query this (full) IMDb data set, we surrender any performance gain made by employing the decomposition.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of synonyms</th>
<th>Average times (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All at once</td>
<td>5</td>
<td>33.72</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>36.10</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>37.14</td>
</tr>
<tr>
<td>One at the time</td>
<td>5</td>
<td>38.95</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>39.09</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>40.16</td>
</tr>
</tbody>
</table>

Table 8.2: Average execution times for two alternative query approaches of vertical decomposition with WordNet
8.2. QUERY OPTIMIZATION AND PERFORMANCE

<table>
<thead>
<tr>
<th>Min. number of Votes</th>
<th>Number of Movies</th>
<th>Query times (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>976.174</td>
<td>54.198</td>
</tr>
<tr>
<td>1</td>
<td>261.749</td>
<td>14.832</td>
</tr>
<tr>
<td>10</td>
<td>141.438</td>
<td>8.136</td>
</tr>
<tr>
<td>25</td>
<td>82.996</td>
<td>4.322</td>
</tr>
<tr>
<td>100</td>
<td>33.386</td>
<td>1.804</td>
</tr>
<tr>
<td>500</td>
<td>11.500</td>
<td>0.677</td>
</tr>
<tr>
<td>1000</td>
<td>7.173</td>
<td>0.486</td>
</tr>
</tbody>
</table>

Table 8.3: IMDb size with respective query execution times

However, because this IMDb subset contains the movies that are mostly requested, we are confident that this decomposition will often suffice. In iFanzy, standardly we give priority to the data set containing popular movies when retrieving information from IMDb. However, if no results were found and the user query indicated a search for movies, we query the bigger set.

8.2.3 Applying available tools, technologies and techniques

To further improve query execution times, additional optimizations were applied. In particular, existing technologies and techniques were evaluated for their usefulness: the use of relational databases, keyword indexing and limiting queries.

Use of relational databases to store RDF data

Where the use of semantic data models offers great possibilities for linking data, current software for storing and manipulating semantic RDF data has its problems when it comes to the performance of very large repositories. One way to recover some of this performance loss is to store well-structured (strongly structured) parts of such large data sets in relational databases. It should be noted that many RDF data storage facilities like Sesame already allow the use of a relational database to store RDF data. However, these relational back-ends are typically very generic, and cannot be configured to store a given part of the RDF data in a specific way.

Relational databases (RDB), originally described in 1970 by Edgar Frank Codd [64], benefit from several decades of research which has led to a very well optimized data management system. Relational databases are very flexible systems as they allow the administrator to tweak and adjust the system to provide the best performance in any given application. Moreover, the administrator has a range of specific optimization techniques at his disposal to further improve query evaluation like e.g. various kinds of indexing, normalization, de-normalization (the process of attempting to optimize the performance of a database by grouping data), etc.

However, when it comes to maintaining Semantic Web data, relational databases are much less suited. This is due to their inherently different perspective. A relational database and SQL are built around the concept of records, consisting of a fixed number of fields with data about a specific resource. If data about this resource resides in different records, they need to be joined. RDF on the other hand, represents a data graph where we can jump from one node to the next by means of triples as representation of every step. Therefore in RDF, we want to be able to query “the graph”, that is, having query expressions that formulate paths through an RDF graph, rather than expressing relations between cells and columns in a relational schema [51]. As an example, in RDF it is very easy to express and query recursive relations like ‘fatherOf’ or ‘isPartOf’. In relational databases however, such relations are hard to model and/or query as they would require a join of the table with itself for every step in the recursion.

Of course, both types of databases have their pros and cons. Relational databases can be fast, even for large data sets, if the data is well-structured and necessary queries can be expressed in straightforward SQL. Databases for RDF graphs preserve the strengths of the structure (like the ability to reason, graph querying, etc.) but tend to become slow when the amounts of triples
CHAPTER 8. DATA RETRIEVAL PERFORMANCE

<table>
<thead>
<tr>
<th>Query</th>
<th>RDF query time (ms)</th>
<th>RDB optimized query time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>648</td>
<td>12</td>
</tr>
<tr>
<td>T2</td>
<td>776</td>
<td>N/A</td>
</tr>
<tr>
<td>T3</td>
<td>896.312</td>
<td>17.360</td>
</tr>
</tbody>
</table>

Table 8.4: Comparison of execution times of an RDF repository with and without relational database optimizations

becomes very large. However, considering these pros and cons, it seems like both approaches can complement each other quite effectively.

To test this idea, we deployed the full IMDb source (65,994,643 triples) in a Sesame repository as well as in a relational database constructed by means of the Java Movie database software\(^5\). However, we adapted the relational data set such that all relevant objects are still uniquely identified by means of the same URIs as in the regular RDF graph. For the first test (T1), we queried each repository asking to retrieve all metadata describing one particular movie. In the RDF graph this means selecting that subgraph describing this movie, while in the RDB it means joining all tables together presenting the same movie information. As can be seen in the first row from Table 8.4, the RDB is here several times faster, due to the specific index which exists of its primary key. Therefore, whenever all metadata for one single IMDb movie is required (like when following a OWL:sameAs property), querying the RDB is favored.

The story changes when inference is required. The RDF database in combination with a reasoning engine can for example provide the correct answer to a question like “Give me all movies produced in Belgium”, pretty fast. In Table 8.4, query T2 shows the average time to retrieve all the URIs of movies produced in Belgium (around 1400). The RDB however, with the schema provided by the Java Movie database, cannot answer it (it would not include movies produced in locations within Belgium). Although it is possible to adapt the RDB so that it can answer the question correctly, it requires extra tables and/or indices. However, in that case it would still only support this particular case of inferencing leading to the fact that the same process needs to be repeated if other types of inferencing should be available as well. Preferably, we use the RDF database to obtain the correct query result directly from the RDF graph.

However, if we now slightly change the query into “Give me all movies produced in Belgium together with their metadata”, both database systems expose their disadvantages. The RDF database would need around 15 minutes to retrieve all the metadata of those 1400 movies, while the RDB is unable to produce the correct set in the first place. However, the combination of both can deliver a both fast and correct result. By first finding all the correct URIs from the RDF database through reasoning, and then for each get the complete metadata description from the RDB, the final result is returned in slightly more than 17 seconds as shown in Table 8.4 by query T3. These results clearly show that combining the advantages of different technologies can lead to better results than each individually could present.

Use of keyword indices

Indices in relational databases are data structures that are constructed to decrease access times to certain parts of the database. Analysis of the four queries introduced at the start of this section showed that the free text search (available to the user in the form of a string search going through all literal properties) is problematic when working with large data sets, since Sesame’s pattern matching facility has a performance linear to the size of the data set. As Sesame does not provide indexing support, we decided to build our own indices specifically to speed up full text search queries. We did so using a relational database. For every program resource, we extracted the literal objects of the properties like e.g. title, keywords, synopsis, actor names, etc. Subsequently, we parsed these literals using white space as a delimiter and filtered them using a stop word filter, retaining only the useful keywords. Every resource and keyword pair was then put in a relational database.

\(^5\)http://www.jmdb.de/
8.3. CONCLUSIONS

<table>
<thead>
<tr>
<th>Query</th>
<th>Without index (ms)</th>
<th>With index (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast keyword query</td>
<td>13.285</td>
<td>223</td>
</tr>
<tr>
<td>Movies keyword query</td>
<td>+1.200.000</td>
<td>3.193</td>
</tr>
</tbody>
</table>

Table 8.5: Execution results for free text search queries with and without index

database table containing an index on the keywords. Every search query issued by the user is first parsed by the Query Refinement component, and then (together with synonyms obtained from WordNet) sent to this relational database. Consequently, the result set of this query is a list of program URIs. The other constraints specified by the user are sent to the relevant Sesame data sets, also resulting in a list of resource URIs. The intersection of these two lists is subsequently returned as the complete result set.

To test the performance gain, we executed two free text search queries, one over all broadcast programs, and the second over the trimmed IMDb data set. For this test both queries were sent to the respective RDF repositories, once with and once without the keyword indices. This technique led to a considerable performance increase for free text search queries, as can be seen in Table 8.5. With this optimization we show that string indexing, although not a standard functionality in RDF repositories, is indispensable where string searches are common.

Use of limit for optimization

Studying the representative queries in the application, we saw that a major optimization could be obtained by the use of limit and offset operators in queries (to both the relational and RDF databases). Indeed, in most cases users do not inspect the whole result set, but only the first few result pages (in which they most likely find what they were searching for). By using a limit clause, the repository/database will be searched for matches until the number of results as specified in the limit clause is found. This can have a significant influence on performance, as the first X results may be found early on in the query execution process, while the whole data set needs to be inspected in order to obtain a complete result set. If the user requests the next page of results an offset is used: the first X of matching results will be discarded by the database and the next Y number of matching results will be returned. Therefore, subsequent result pages will be more expensive to calculate. However, as this is generally not the default behavior of users, this should not reduce the overall performance. The test results in Table 8.6 illustrate the performance gain by including a limit clause (with increasing limit operands) in a typical iFanzy query executed on the IMDb data set.

<table>
<thead>
<tr>
<th>Limit</th>
<th>Average execution time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>595</td>
</tr>
<tr>
<td>100</td>
<td>628</td>
</tr>
<tr>
<td>1.000</td>
<td>834</td>
</tr>
<tr>
<td>No Limit</td>
<td>1.665</td>
</tr>
</tbody>
</table>

Table 8.6: Average execution times for increasingly limited queries

8.3 Conclusions

In this chapter we discussed the performance of a search action following our approach towards personalized access. Given the size of the RDF data graph and the complexity of various reasoning techniques, the search performance was, as expected, too slow to feed a responsive end-user application. However, through a number of known and generally applicable techniques like splitting the data sets (both horizontally and vertically), using different types of databases, building custom keyword indices, enabling parallelization of queries and focussing on the most popular data
subsets, we managed to improve the performance considerably, in comparison to a naive approach involving one large data repository containing all the schema as well as instance data. However, these measures also demand a more specialized query handling, responsible to make sure that the overall database behavior is identical to what would be displayed by one big central database. This includes for example regulating the query pipeline which determines the execution order of queries, and the joining or intersecting of different query results.

<table>
<thead>
<tr>
<th>Query</th>
<th>Query times before optimizations (ms)</th>
<th>Query times after optimizations (ms)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>31.526</td>
<td>1.519</td>
<td>21x</td>
</tr>
<tr>
<td>Q2</td>
<td>138.354</td>
<td>1.049</td>
<td>132x</td>
</tr>
<tr>
<td>Q3</td>
<td>224.663</td>
<td>2.729</td>
<td>82x</td>
</tr>
<tr>
<td>Q4</td>
<td>+1.200.000</td>
<td>8.895</td>
<td>&gt;135x</td>
</tr>
</tbody>
</table>

Table 8.7: Average execution times of typical iFanzy queries before and after optimizations

To show the final cumulative improvement of all optimizations together, we re-executed the initial four representative queries we discussed at the start of this chapter. In Table 8.7 we see for those four queries both the original execution times (achieved on one big repository without any optimization) as well as the final query times after applying all optimizations discussed. The improvements are significant without making any functional concessions. With these optimization techniques applied on our approach, we answered research question 7 (Given our approach towards user-adapted data retrieval, how can we optimize data retrieval efficiency in terms of querying speed?).
Chapter 9

Interfaces and Interaction

The idea behind iFanzy was originally conceived by Stoneroos Interactive Television\(^1\) for the Passepartout project which started beginning 2005 and included various companies like Philips, Thomson, ETRI and knowledge institutes like the Technical University Eindhoven. The project was aimed at coupling home media-centers to home networks for rendering scalable and personalized content from high definition television (HD-TV) to lower definitions (mobiles) in a seamless fashion. The content had to originate from different sources among which the, back then, highly anticipated Blu-ray disk played a central role. From the start, there was a strong emphasis on a personalization engine which would among others provide input for a Personalized Electronic Program Guide (PEPG) built around the novel concepts of Blu-ray disks and the TV-Anytime Packaging concept. Due to the expertise of the TU/e in terms of personalization and Semantic Web research, the TU/e and Stoneroos found each other within the project to start the development of such a PEPG. The first version of a TV personalization engine was initially called Blu-IS, pronounced as “Blu-ICE”, which represented the Blu-ray Information System. Blu-IS was the technology responsible to manage information streams coming from different sources like a Blu-ray disk, DVD, PVR, etc. and secondly to model the user to be able to personalize the TV experience. iFanzy was initially described as a set-top box application responsible for creating personal dynamic filters for incoming content. Such a personal filter could then for example turn a regular EPG into a PEPG. iFanzy was thereby supported by technologies like Blu-IS and Blu-ray.

Later, the desire came to broaden the horizon of Blu-IS, making it a more generally applicable approach. By investigating data modeling, broaden user model structures, allow intelligent searching, provide content recommendations, evaluate the performance and allow the user to interact with such a system, we described a general approach for personalized and context-sensitive data retrieval. This approach however, slowly became larger and richer than initially required from the Blu-IS architecture. Therefore we renamed Blu-IS into SenSee, which was an abbreviation for “SENsing the user and SEEing/serving the content”. SenSee still fulfilled the requirements initially set in the Passepartout project, but could also do more. SenSee became the first implementation of the architecture built according to our initial approach. From this initial approach we later extracted the approach elucidated in Chapter 4.

After the official end of the Passepartout project, together with Stoneroos, the work and development on iFanzy and SenSee continued. However, for the sake of consistency and branding our SenSee server was renamed, this time straightforwardly into the iFanzy Server. The name ‘iFanzy’ itself, now refers to all client applications developed on different platforms and running on top of the iFanzy Server. Recently, iFanzy was launched commercially and picked up by many journalists [158, 227, 98, 69].

In this chapter we introduce the iFanzy system from a more pragmatic perspective, and show how we implemented the techniques and approach discussed in previous chapters. We show the architecture of the iFanzy system, both in terms of the server and the clients, and illustrate how

\(^1\)http://www.stoneroos.nl/
they work together. Afterwards, we go through the three iFanzy clients and show how the user can interact with them. Lastly, we evaluate the Web interface by means of a user study. This chapter addresses research question 8 (How can the user interact with such a system, following our proposed approach and applied in the television domain, effectively?)

9.1 Introduction

In this thesis so far, we emphasized the unbridled growth of various data sources, new relations between those sources, Internet traffic and connectivity numerous times. Moreover, new forms of media are emerging as digital systems are converging. Different content, e.g. from TV, radio, music, homemade images and videos, are no longer bound to separate devices or to local storage. Furthermore, people are, through the newest generation of gadgets, gaining the ability to obtain any information at any time. As envisioned by for instance IBM [33], future media will become more pervasive and offer a more ubiquitous and immersive experience, as increasing technological sophistication brings new media environments. This evolution has, among others, led to the concept of the context-aware connected home where sensors, media containers, wireless network technology, media consumption platforms and ubiquitous devices immerse the user in one single but consistent media experience. Many companies, like for example Philips, are picking up these new possibilities. In the Philips HomeLab, a simulation of the household of the future, scenarios of Ambient Intelligence are implemented and tested [74]. In [168], an overview of research in context-aware connected homes is presented.

Following these evolutions, in a strong collaboration between the TU/e and Stoneroos, several interfaces were built to provide a seamless and ubiquitous platform-independent television experience to the user. In this chapter, we discuss some of these interfaces which were constructed consecutively in a number of steps, finally leading to the latest: the iFanzy iPhone application.

In Section 9.2, we start this chapter with a short introduction of the back-end server. Just like the interfaces, this server slowly evolved from the initial version, called SenSee, into the currently employed iFanzy Server. This server forms the beating heart of the application, and is a direct implementation of the approach described in Chapter 4.

As previously explained, initially one of our first steps towards a personalized television experience was the SenSee framework. Therefore in Section 9.3, we describe the SenSee Web application, our first interface towards the personalized retrieval of TV data. This Web application uses state-of-the-art technologies like Google Web toolkit and Ajax to present its functionalities responsively. SenSee, including this Web application, was nominated for the Semantic Web Challenge at the International Semantic Web Conference in 2007 (ISWC’07) where it ended as runner up [24].

Later, iFanzy replaced the SenSee framework, and therefore we take a closer look at the current iFanzy interfaces in Section 9.4. Presently, iFanzy consists of a Web site, a set-top box interface and an iPhone application. Each of these commercially employed platforms is carefully tailored to support the user with the functionality mostly expected from the respective platform. Behind the scenes, all three connect to the same server assuring their mutual synchronization of data, thereby guaranteeing that every action performed on any of the platforms, has an immediate effect on all.

9.2 The back-end server

As previously explained, the back-end server went through a number of steps from the server supporting the SenSee framework until the current iFanzy Server. However at the core, the general lines and ideas behind the approach of the server remained the same throughout its evolution. Therefore, the back-end server discussed in this section represents the latest incarnation of the server, which is also currently employed in the iFanzy architecture. In general, this server is the direct implementation of the approach described in Chapter 4, and responsible for the integration of program metadata coming from different heterogeneous sources, the maintenance of all user models and the logic to provide user-adapted data retrieval.
In Figure 9.1, we see the general outline of the architecture of the server roughly consisting of four separated parts. Firstly, at the bottom we see different sources of content (or at least the metadata describing it). Among these we see IP sources (e.g. theme channels, trailers), regular broadcast, local PVR storage and portable media like DVD or Blu-ray disks. In principle every potential source can be added there, at least when providing proper descriptive metadata. Secondly, in the figure’s middle, we see the layout of the iFanzy server. The server is modeled as four different interconnected layers where each adds a level of abstraction. The retrieval layer obtains and integrates the metadata, the package layer handles the querying process, the personalization layer adapts the content retrieval to the user and the administration layer focusses on the client-server communication and administrative tasks. Thirdly, at the left hand side of the figure we see a list of external services which are used by the iFanzy server architecture. Among these services we see some TV-Anytime specific services (like the CRID Authority and the Metadata Service), but also for example the User Model, Ontology and Filter Service maintaining user models, ontological information and various metadata filters respectively. Fourthly and finally, at the top of the figure we see a series of client applications which utilize the functionality provided by the iFanzy server.

In this architecture some components, like for example the User Model Service (UMS) which manages the user models as described in Chapter 6), are modeled as a service rather than a separate component within the architecture. The main reason to work with services lies in the fact that we want to grant other parties access to these services. Imagine for example that a different institute or company creates an application to parse user data from the Facebook network. With the UMS as a service they can directly add user data by means of the UMS interface, immediately allowing the iFanzy server to directly benefit from this additional information.
CHAPTER 9. INTERFACES AND INTERACTION

9.2.1 Content Retrieval Layer

Within the server’s architecture, the **Content Retrieval Layer** is responsible for retrieving metadata from different sources and integrate it into a consistent data graph as described in Chapter 5. In Chapter 5 we also saw that we deal with two different kinds of sources, namely *root sources* and *enrichment sources*. Where the former tries to retrieve the television program schedules for different channels (or program catalogues in case of VOD collections), the latter tries to enrich the individual program descriptions as much as possible. In Figure 9.1 we see that the main components of this layer facilitate metadata retrieval, transformation and integration before everything is stored in the database. The Database Management System to which it connects uses Sesame, an open source framework for storage, inferencing and querying of RDF data [52].

In iFanzy we always try to maintain program metadata of at least the seven upcoming days. Such a sliding window, which is updated daily, provides on the one hand the necessary buffer in case some of the sources are temporarily down, and on the other hand for the user a complete week of overview of programs to come. Therefore, every night the retrieval component runs an update to retrieve new data from the root sources. The two major root sources in iFanzy are BBC Backstage and XML-TV grabbers. The BBC Backstage Web site is daily updated with retrievable program schedules (even nine days in the future), while the XML-TV data has to be retrieved via a Perl script. When working with different root sources, it is always possible that different sources retrieve data for the same channels. E.g. both the XML-TV UK grabber and BBC Backstage contain metadata for the channel “BBC One”, “BBC Two”, etc. In such a case, the OWL:sameAs property is used to connect instances describing the same programs. As described in Chapter 5, sometimes metadata needs to be transformed before it is integrated in iFanzy's data graph. After the update of the program schedule, the server checks for every program whether it can further enrich the program by means of any of the included enrichment sources like IMDb, VideoDetective, DBpedia, etc.

9.2.2 Querying and Clustering Layer

The **Querying and Clustering Layer** is the next layer on top of the Content Retrieval Layer. This layer's goal is twofold, firstly it is responsible for dealing with queries generated by the Query Refinement component, and secondly it facilitates the optional creation of TV-Anytime packages as described in Section 3.4.3, utilizing functionality of the CRID Authority (CA) and the Metadata Service (MS) as described in the TV-Anytime specification and briefly touched in Section 3.4.

Besides a link to the CRID Authority, the Metadata Service and the Database Management System, the layer has two main subcomponents, shown in Figure 9.1. Firstly, the Query Handler processes all incoming queries necessary to support the personalization strategies described in Chapter 7. After the query refinement component parsed the initial user request, the resulting query is sent to the query handler component. Here the query is split and adapted to match the databases following the optimization steps described in Chapter 8. E.g. whenever a data set is decomposed, either vertically or horizontally, the query needs to be split accordingly or to be sent to several repositories respectively. As a consequence, the query handler is also responsible for combining the results returned from those different repositories.

The second component in this layer is the Package Creator which can “on-demand” create a package out of retrieved results. Originally, the TV-Anytime packaging strategy was created to allow content producers and/or broadcasters to create a TV-Anytime packages which bundles different but related content elements together. For example a “Lord Of The Rings” package could include the three movies, a making of, three soundtracks, a bunch of pictures and some small games. These parts of the package are then put together in a hierarchical tree structure consisting of components (content references in the tree) and items (nodes in the tree), previously shown in Section 3.4.3. Therefore, packages were not originally meant to be created automatically, but merely as a more advanced content clustering technology. However, even when created automatically, packages can provide a surplus value to the user in terms of navigation, overview and inter-content relations. We implemented some simple but illustrative strategies to accomplish au-
tomatic package generation. If the user for example searches for a series like for example *Friends*,
the Package Creator will create a package containing a different node (item) for every season of
the series, such that episodes of the same season are grouped together. However, such a strategy
is very specific and can only work for particular cases. A more general approach is very similar to
*Faceted Browsing* which was originally described in [262], and afterwards adopted by the Semantic
Web community [124]. In this case, we cluster the results, based on the value of some well-chosen
program properties like for example channel or genre. Searching for ‘universe’ could then for
example result in a package where you have a node “science-fiction”, ‘action’, “nature sciences”,
etc., where each of those nodes then represents the group of programs which were annotated with
the corresponding genre and matched the keyword ‘universe’.

### 9.2.3 Personalization Layer

The Package Handling Layer enables us to query the metadata retrieved in the Content Retrieval
Layer, and returns the results either as an unordered list or as a package of items and components.
This result is then returned to the *Personalization Layer* which issued the original request after
refinement. As discussed in Chapter 7, this result is now further filtered to obtain the final and
personalized result. However, to do so it needs the knowledge from the three external services
which are made available through the *Services and Model Manager*. In Figure 9.1 we see the *User
Model Service*, the *Ontology Service* and the *Filter Service* at the left hand side. Note that these
same three services were previously introduced in Figure 7.1.

The User Model Service (UMS) is responsible to maintain the user model of every iFanzy user.
It catches generated events, deduces new information, tries to counter cold start problems and
deals with the user’s context as described in Chapter 6. Which events, both implicit and explicit,
it can catch from the different iFanzy interfaces, we explain further down in this chapter.

The Ontology Service (OS) is basically the central repository for ontological information. In
this repository we store all the ontologies, vocabularies and thesauri which are used during the
personalization process. Every time we need knowledge of concepts, properties or hierarchies, we
can use specialized calls in the API of the service. We can for example ask for all synonyms of the
word ‘bike’, which would return among others the word ‘bicycle’. We can also for example ask for
the range (via *rdfs:range*) of a property to interpret its value. The OS is also responsible for the
keyword conceptualization step like discussed in Section 7.2.1.

The last service is the Filter Service (FS) which is responsible to maintain a set of rules which
can be used to filter or rank results in the query result set. The effective filtering and ranking is
done by the Content Filtering component in the Personalization Layer, however, the FS maintains
the rules.

### 9.2.4 Server Administration Layer

The last layer in the iFanzy server architecture is the *Server Administration Layer*, as shown in
Figure 9.1. This layer’s main responsibility involves the communication between the different
clients and the server. It exposes an API which enables functionality like searching, getting rec-
ommendations, etc. Besides the API provisioning, this layer roughly contains two other important
components. Firstly we have the User Identification component, which allows a client to register
on the iFanzy system, and later to authenticate. Besides a regular login/password combination,
also RF-ID identification (which could be handy in a living room context) can be used. Secondly,
the layer facilitates a Context Detection component. All the actions performed by the user (like
rating a program, setting a reminder, changing preferences, etc.), generate events which enter the
system via a system’s call in the API. This feedback is then sent to the UMS where it is saved as
an event in the user profile (as described in Chapter 6). However, before sending it, the Context
Detection component checks whether the action has an associated context. Although determining
context is normally a client responsibility, if no information is present, this component can fill in
some holes like e.g. the device or location the action was performed from (based on device id or
IP), at which time the action was performed, the current audience at the device (in case more
people authenticated themselves from that IP), etc. Do note that user feedback could also be sent directly to the UMS via the UMS API (because it operates as an independent service). However, the UMS does not always have the same information to automatically determine context that the server has. The server for example can keep track of which people are currently logged-in (e.g. via the same set-top box), while the UMS cannot.

9.3 SenSee

Our approach described in Chapter 4, facilitates personalized access to semantically annotated data sources. However, until now we have only spoken about the techniques utilized behind the scene to allow such access, and not about how a user can benefit from these techniques. In this section, we discuss the first illustrative interface built within the Passepartout project, allowing the user to interact with the system described so far. This interface, which was part of the SenSee framework, was developed as a Web application trying to provide personalized access and search to large sets of television programs.

The Web application we created to demonstrate our approach, was built using state-of-the-art techniques and tools in the Web development world. For example, by utilizing the Google Web Toolkit2 (GWT) it was possible to build a Web application that has the richness and flexibility of a regular Java application, while it preserves the look and feel of a contemporary Web page. To facilitate this, GWT provides a Java API which enables programming without the usual HTML/JavaScript difficulties and common browser incompatibilities. Afterwards, GWT compiles the classes and dynamically converts them to regular JavaScript. By doing so, the Web application automatically appears as a regular Web site using standard Web techniques, however, also already fully featuring the Ajax (short for Asynchronous JavaScript and XML) technique, facilitating asynchronous method calls. This aspect of Ajax is particularly important, because it allows us to retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. E.g. if the user is browsing through a list of twenty programs returned after a search request, in the background the query for the next twenty results was already executed when the user neared the end of the initial list. Upon return, results from the second query are then added seamlessly to the initial result.

![Figure 9.2: The client search interface](http://code.google.com/webtoolkit/)

2http://code.google.com/webtoolkit/
Figure 9.2 shows a screen shot of the Web application’s search interface. With this panel a user, or a group of users can search for television content. At the top right we see that three users are currently logged-in. Users can log-in via a separate login panel (reachable by a link at the top left) and log-out (by clicking on their user icon) at any time. Underneath the user icon bar, we see a text input field to fire a keyword query. Further, on the left, we present three well-chosen facets or dimensions, including time, genre and location, which the user can utilize to construct a data request. Time is visualized as a calendar showing the current day (here the 12th of December), and the following six days. Besides selecting a day, a user can further specify a time frame. Time frames are indicated by their starting hour (for example ‘15h’ indicates the time frame ‘afternoon’), and can be found at the bottom of the calendar widget. On-demand content, as opposed to programs which are broadcast, is not time-bound and will therefore not be affect by a time-selection. The Genre facet is visualized as a tree, where each node (genre) can fold out showing sub-genres. The tree is built based on the SKOS:broader and SKOS:narrower relations.

Users can select one or more genre restrictions for the program they are searching for. The Location facet basically visualizes a large location tree based on the GeoNames data set. Every node is a location, and when unfolding we see all the locations which have a GEO:parentFeature property to the original location. Similarly in [115], Harth presents VisiNav, another interface to search and navigate RDF data sets collected from the open Web. VisiNav is similar in that regard that they also combine keyword search with selectable objects from the Semantic Web. In our application, with these four means of restricting a query (keyword input and three dimensions) the user can for example ask: “Give me all programs featuring “Brad Pitt” available tomorrow evening with genre ‘Action’ and produced in the USA”. This query can be formed by putting the string “Brad Pitt” in the text input field, selecting the 13th of December (if today is the 12th of December, this denotes ‘tomorrow’) and clicking ‘18h’ for the day frame (‘evening’) in the calendar, the genre ‘Action’ in the genre tree and the location ‘USA’ in the location tree. With these actions, the user, maybe without realizing it, conceptualizes the query by selecting the correct concepts in the time, genre and location dimensions. It would also have been possible for him or her to type the entire query textually in the keyword field. In this case, the query refinement algorithm will automatically try to conceptualize every substring, before sending it to the database.

In Figure 9.3 we see the SPARQL query to which the previous user query eventually would get translated. In this query we see that the string “Brad Pitt” was recognized as a person and should appear in the credits of the program. The user however did not specify whether he should be appearing as actor, producer, etc., so no role is specified. Further, we see that this program should have the genre ‘Action’ (IFZ:3.4.6) and be produced in the ‘USA’ (GED:6252001). However, due to our inferencing in our knowledge base, programs with genres narrower than ‘Action’ (like ‘Horror’ (IFZ:3.4.6.6)) and/or programs produced in a place located within the ‘USA’, are automatically included in the result of the query.

In Figure 9.2 at the bottom left, we see some checkboxes to set more advanced application preferences. Here we can for example indicate whether we want the user’s model to be taken into account, whether we want to include context, if the knowledge base should enable inference, etc. The option controlling whether or not to take the user model into account is particular handy because it can show the difference between a ‘normal’ and a personalized search. When disabled, user model information is not included at all, just returning all the results matching the original user request. The switch controlling the inclusion of context is only available when user model information is taken into account. However, the differences can still be significant. When context is disabled, the data retrieval process uses the average of all user likings for a specific concept. So if the derived user liking for the genre ‘Action’ equals 8 in the evening and 4 in the morning (on a scale from 0 to 10), 6 will be used when the inclusion of context is disabled. Otherwise, the liking matching the current context best, is used. Via these tabs in the user interface, we can influence and fine-tune the content retrieval process and evaluate its results.

Via the Content Tree View (in the middle of the figure), we can browse returned results which are here depicted as leafs in a tree. This tree is build based on the three dimensions (time, genre and location) in an order of choosing. Currently, the ‘genre’ dimension was prioritized and thus results are ordered by their genre. Further, this view also shows a “Why are these recommended”
link, which provides an overview of on one hand all the semantic relations exploited by the reasoner and on the other all additional steps performed in the query refinement process, which were used to come to this result. If the user for example searches for the genre ‘Action’, and a horror movie is returned, this link will explain that the genre ‘Horror’ has a SKOS:broad relation to the genre ‘Action’ leading to its inclusion. Similarly, if a program matched with the keyword ‘bicycle’ while the original keyword was ‘bike’, the WordNet synonymity relation between these two keywords is shown.

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX ifz:<http://www.ifanzy.nl/>
PREFIX time:<http://www.w3.org/2006/time#>
PREFIX geo:<http://sws.geonames.org/>
PREFIX mpeg7:<urn:mpeg:mpeg7:schema:2005-03>

SELECT ?program
WHERE
{
  ?program rdf:type ifz:ProgramInformationType .
  ?program ifz:programId ?ID .
  ?program ifz:BasicDescription ?basDesc .
  ?basDesc ifz:genre ifz:3.4.6 .
  ?basDesc ifz:ProductionLocation ?loc .
  ?loc geo:parentFeature geo:6252001 .
  ?basDesc ifz:CreditsList ?cList .
  ?cList ifz:CreditsItem ?cItem .
  ?cItem ifz:role ?role .
  ?cItem ifz:personName ?pName .
  ?pName mpeg7:givenName ?name .
  FILTER(?name = "Brad Pitt")

  ?schedule rdf:type ifz:ScheduleEventType .
  ?schedule ifz:PublishedStartTime ?time .
  ?time time:hour ?hour .
  ?time time:day ?day .
  ?time time:month ?month .
  ?time time:year ?year .
  FILTER(?day = "13" && ?month = "12" && ?year = "2009" && ?hour >= "18" && ?hour < "22")
}
```

Figure 9.3: Example of a SPARQL query fetching television programs

Whenever a program is clicked by the user, a new page is opened showing all the information we have retrieved from different enrichment sources for that program. In Figure 9.4 we see this page for the program “Lord Of The Rings: The Two Towers”. On the left we see all textual metadata like ‘Director’, ‘Actors’, ‘Keywords’, ‘Plot’, etc. and on the right we see a VideoDetective trailer playing. Close to the bottom, we see a picture gallery which can be clicked to see each one at full resolution. At the bottom of Figure 9.4, we see some button to close the pane, but more interestingly, also to add the program to the user’s favorites selection, or to rate it by means of five stars. This rating action directly influences the user model of the currently logged-in user(s).
9.4. THE IFANZy USER INTERFACES

With the switch from the SenSee framework to iFanzy, the architecture was improved and further elaborated. Moreover, trying to evolve towards a truly ubiquitous environment, we realized we needed to broaden the horizon and promote iFanzy on different complementary platforms. By doing so we could offer the user, next to an Internet portal where he or she can check program schedules and perform a multitude of actions, an interface on a set-top box to use when watching television and a mobile application to quickly check television related information whenever none of the other platforms are available nearby. In the following three subsection, we give an overview of current iFanzy interfaces.

9.4.1 The iFanzy Web site

In Figure 9.5 we see the interface of the iFanzy Web page\textsuperscript{3}, when a user is logged-in. The site basically consists of five tabs: ‘Home’, ‘TV-Guide’, ‘My TV-Guide’, ‘My Profile’ and ‘Search’. Centrally on the home page at the right hand side, we see three movies which will be broadcast the coming hours and fit best to the currently logged-in user. For the ‘top’ movie (in this case “The House Of Sand”) also a trailer is shown to give the user a quick preview of the best matching movie. On the bottom left, we see a small personalized EPG (PEPG) listing three channels currently showing all programs broadcast between 15:00 and 17:30. The programs shown on these channels get a specific color ranging from very light orange toward very dark, indicating how well these programs fit with the profile of the current user. In this way the interface provides a recommendation for the programs playing right now. For a more elaborate overview of channels

\textsuperscript{3}http://www.ifanzy.nl
CHAPTER 9. INTERFACES AND INTERACTION

and programs, the interface also contains a complete listing which can be found under the tab “TV-Guide” showing all the channels available in iFanzy, colored accordingly in shades of orange. However, the small PEPG on the home page depends on the current user as it shows his or her three most appreciated channels, either indicated explicitly by the user or deduced automatically from the user model. To get an overview of the user’s personal preferences, we provide the “My Profile” tab where a user can edit his or her settings. Here, the user can also include extra or exclude unwanted channels to be shown in the PEPG. On the home page, next to the PEPG, on the right, we show a search bar to look for specific content accompanied by a tag cloud showing the most popular search terms. The iFanzy search takes various semantic relations into account, as explained in detail in Section 7.2. For a more elaborate and fine-grained search, the user can click the ‘Search’ tab at the top of the page. Here, the user can restrict a search action for example by means of a set of genres and/or a time frame selection, in a similar way first shown in the SenSee interface.

![Image of the iFanzy Home page](image_url)

Figure 9.5: The iFanzy Home page (Stoneroos Design Team)

Next to the tab “TV-Guide” (which shows the PEPG including all available/selected channels), we also have the tab “My TV-Guide”. Under this tab we can find a personal selection, made by the user himself, providing an overview of programs he or she would like to watch. This feature represents the digital form of people manually marking programs in a paper EPG, they definitely want to watch. Via this feature people can make a program selection at work (e.g. while taking a break), and print it to get a nice overview. This functionality enables maximal enjoyment and quality time at home and prevents the user from any further searching, during that time.

Whenever a program is clicked, either in the condensed or complete PEPG, an info-pane slides down like shown in Figure 9.6. In this figure, the details of the program “How to Deal”, broadcast on ‘RTL8’, are shown. At the top left, we see a rating bar which can be used by the user to indicate how much he or she likes this program. Standardly, it shows how iFanzy predicts the user likes it (currently 8 on a scale from 0 to 10). Note that this bar uses the same shades of orange as used in the PEPG. Underneath the rating bar we see the VideoDetective player showing the program’s trailer. Below the trailer we show a part of the program’s metadata including the channel and time of broadcast, the genre (here ‘drama’, ‘movie’, ‘comedy’ and ‘romance’) and a short synopsis. At the top right, we see a list of all people involved in this program together with their role.
this case, five main actors and the director of the movie are listed. At the bottom right we provide a link to the corresponding IMDb page, in case the user requires further background information.

At the bottom left, we see five icons which represent actions which the user can perform on any particular program. The first icon leads to the addition of this program to the user’s personal selection (available under the “My TV-Guide” tab). The second one can be used to set this program as a favorite. The third is the record icon by which the user can schedule a program to be recorded on his set-top box at home. Do note that this functionality depends on various external factors like the available set-top box, Internet connectivity and whether iFanzy is running on the box. With the fourth icon, the user can set a reminder which is sent to him either by mail or by SMS, a specified time before the program starts. The fifth and final icon allows the user to recommend a program to a friend who might be interested. The tip can be send via the iFanzy network if the recipient already has an iFanzy account, or via regular mail otherwise. As explained in Chapter 6, actions triggered by these buttons cause more than the functionality they represent. Behind the scenes, pressing any of these buttons, fires an event which is caught in the user model. Implicit events, which are gathered in this way, are maintained in the event model and can, after closer examination, lead to the materialization of an entry in the user model. If the user, for example, sets a program reminder every Tuesday for the new episode of Friends, data mining algorithms can catch this pattern and materialize the user’s interest in this series, genre, etc. In case of explicit events, like e.g. rating a program, this process is omitted because it represents an unambiguous user statement.

9.4.2 The set-top box interface

One of the main objectives of iFanzy is to provide program recommendations and intelligent search via a set-top box in the living room. Therefore, in a number of steps, an interface has been developed for a specific set-top box which is currently being tested by around 5000 families in the Dutch village of Hillegom. In Figure 9.7 we see the welcoming screen of this interface. Important here is that the STB interface is always tailored to match the properties (e.g. resolution, margins, etc.) of a regular television screen. Although more and more people are adopting High Definition screens, we also must accommodate people who have not.
In essence, the functionality on the STB interface is similar to that on the Web. However, it has been tailored to match any of the advantages and peculiarities of the medium. Considering for example the limited means of a remote control to interact with the television set, at the bottom of the screen we see that specific iFanzy actions are coupled to the well-known colored buttons (red, green, yellow and blue) on the remote control, for the user’s convenience.

Figure 9.7: The iFanzy set-top box login screen (Stoneroos Design Team)

Figure 9.8: The iFanzy set-top box PEPG screen (Stoneroos Design Team)
9.4. THE IFANZY USER INTERFACES

Taking a quick look at the interface, centrally in Figure 9.7 we can see a series of buttons: “Add Profile”, “TV Guide”, a list of already available profiles (here ‘Erik’ and ‘Geert’) and “New Profile”. Straightforwardly, “TV Guide” brings you to the regular EPG, “New Profile” creates a new iFanzy profile and pressing any of the already available profiles (e.g. ‘Erik’) allows you to log in as that person after entering the correct password. Any new account created via the box also immediately becomes a valid account on the Web. Manifesting iFanzy as a truly integrated and ubiquitous application, the same person using different interfaces must internally be connected to the same iFanzy user. Therefore, the “Add Profile” button allows a person who already has an iFanzy profile (e.g. created via the Web site), to add it to this particular set-top box.

At the top of Figure 9.7, we see the navigation bar providing the following buttons: “Home page”, ‘PEPG’, ‘EPG’, ‘Search’, ‘Settings’, “Change user” and ‘Exit’. Currently, this interface does not yet support logging-in with more than one user at the same time. Only when a user logs in, the PEPS becomes available. In Figure 9.8 we see the PEPS generated for user ‘Tjeerd’. Similarly to the front page of the Web interface, we see the user’s three favorite channels listed (here ‘NED1’, ‘Veronica’ and ‘Discovery’). Also similarly, programs receive an orange shade indicating how strongly they match this user’s profile. However, because of the resolution limits, it is less convenient to show a large channel overview like we have in the Web interface. Therefore, to guide the user to the interesting bits, we show the three top recommendations playing now and the three top recommendations playing later across all channels. Further, a button “More recommendations” allows the user to obtain the next six. At the top we see the program metadata of the currently selected program (in this case ‘Friends’), among which we see a genre, start- and end-time, channel and a short synopsis. Being one of the advantages of the STB, at the right hand side of the program’s metadata we see a small window with a live feed of the program, or a trailer when the program did not start yet. At the bottom of the page we see again the well-known colored buttons to navigate through the PEPS.

9.4.3 The iPhone application

To make iFanzy truly ubiquitous, a mobile application is indispensable. After all, people carry their mobile everywhere around like a personal buddy, guaranteeing permanent Internet connectivity. Checking your mail in the train, looking for directions on Google maps during a hike, quickly checking the closing hour of the supermarket in the car, etc., the possibilities are limitless. Furthermore, mobile devices are growing more powerful by the year. Live Internet access, large touch screens, plenty of memory and a powerful CPU, have turned them into pocket-size mobile computers. In the same regard, it seems likely that people want to check the coming evening’s TV schedule, for example in the train on their way home or want to rate some programs to improve their user model when having a spare minute.

The iPhone iFanzy client tries to bring the iFanzy experience to the mobile phone. To do so, it runs on top of the available iFanzy server through which it also automatically runs in sync with the other two clients. In our regard, the iPhone interface should just enable people to quickly see what is on television tonight. In Figure 9.9 we see the iFanzy interface on the iPhone. In this interface the user can select favorite channels, favorite genres and rate programs. After doing so, the user gets a quick overview of programs available either today or tomorrow, together with an indication how well every program fits their profile. Further, on the next page general program recommendations can be seen to plan your coming TV evening. Again, once an iPhone profile is created, is also becomes valid on the other platforms and, also the other way around, if the user already had an account, it can be loaded on the iPhone. However, the nicest benefit comes from the fact that users can rate programs on their iPhone, and later in the evening, when they turn on their TV/STB combination, the recommender system already took their latest input into account. This cross-platform integration is what makes the iFanzy a truly ubiquitous and immersive television experience.
9.4.4 iFanzy interface evaluation

Performing a user interface evaluation is not an easy task. The field of Human Computer Interaction (HCI) studies the interaction between a user and computers (both in terms of software and hardware), facilitated through some kind of interface, in its broadest form. The goal of HCI research on a given interface is to improve the interactions between users and computers by making computers more usable and receptive to the user’s needs [254]. However, the fact that an interface needs to be made more usable can have many different reasons. An interface can lack responsiveness, the choice of colors can be eye-tiring, functionality cannot be found or used straightforwardly, interfaces can be confusing, etc. Therefore, HCI focusses on the entire user interface development process including design, implementation and evaluation until user satisfaction becomes sufficient.

An independent but very related field is that of usability. Usability is concerned with investigating and improving the efficiency, effectiveness and safety of utilization, the ease of learning and the satisfaction of using any particular tool or object. While the user experience of an interface is a very subjective matter, usability strives to be objective and can therefore be measured to some extent. We can for example measure exactly how many and which steps are necessary to obtain a cup of coffee from the vending machine, and how long on average every step takes.

A usability evaluation can be performed by means of many different evaluation methods each with advantages and disadvantages (for more information we would like to refer to [183]). To perform a usability evaluation on iFanzy, we decided to employ three different methods including a Cognitive Walkthrough, the Thinking Aloud Method and a Heuristic Evaluation.
For the test itself, we let a group of 60 students (working in groups of three) perform each of these tests on an arbitrary number of freely chosen test subjects. Do note that to perform these tests on iFanzy, we chose to constrain ourselves only to the Web interface, since this platform is the easiest to approach for test subjects.

**Cognitive Walkthrough**

The Cognitive Walkthrough [152] is a usability inspection method without users, that focuses on evaluating a design for ease of learning, particularly by exploration [251]. The idea is that, instead of giving a formal training of a software package, users learn by using the software when they need it. In a Cognitive Walkthrough, reviewers evaluate a proposed interface in the context of one or more specific user tasks. As a consequence, every step necessary to perform a task can be evaluated, attempting to uncover design errors that could interfere with learning by exploration. Reviewers have to carefully consider the target population of users who are eventually going to use the interface, since differences between the reviewers and the target users can be significant. The Cognitive Walkthrough method is very good way to refine requirements. Moreover, a fully functional prototype is not necessary to perform it, a complete design and specification is.

As mentioned in [251], for each task there must be a description of how the user is expected to view the task before learning the interface. Then the reviewer performs the task guided by four criteria which need to be checked for every step part of the task. These four criteria are:

- Will the user try to achieve the right effect?
- Will the user notice that the correct action is available?
- Will the user associate the correct action with the effect they are trying to achieve?
- If the correct action is performed, will the user see that progress is being made towards solution of their task?

In the test, different student groups could choose different iFanzy-specific tasks to use during the Cognitive Walkthrough. Among the tasks evaluated by the students we see for example “Set a reminder for a program”, “Add a program to your favorites”, “Change the ordering of your channels”, etc.

**Thinking Aloud Method**

Thinking aloud [151, 91] is a method that requires subjects to talk aloud while solving a problem or performing a task. This method traditionally had applications in psychological and educational research on cognitive processes, but also for the knowledge acquisition in the context of building knowledge-based computer systems [139]. Such a method can be used to analyze in detail the way in which humans perform tasks, mostly in interaction with a prototype computer system. The ultimate goal is to develop end versions of computer systems that map on users tasks and strategies in performing these tasks and trigger the cognitive dynamics of the user in such a way that the intended tasks can be accomplished with minimal cognitive effort [139]. This method is often popular because it is reasonably cheap and the results are close to how a user really experiences the task.

To perform such a test we for example ask the users: “Tell me what you are thinking about when you perform task A”. Naturally, it is important that the user is encouraged to talk while he or she acts, to capture every sign of hesitation, confusion or misunderstanding. Afterwards, results can be compared with how the designer intended the interface to be used, potentially leading to rethinking it.

For the thinking aloud test in the iFanzy interface, students had to find an independent subject willing to execute a simple task, while exclaiming everything coming up during the process. Among the goals chosen by the students we can find: “Find a program playing tomorrow”, “Record your favorite program”, “Indicate which program you have seen yesterday”, “Rate your five most preferred programs”, etc.
Heuristic Evaluation

Heuristic evaluation is an informal method of usability analysis where a number of evaluators are presented with an interface design and asked to comment on it [184]. In other words, the users or evaluators are asked to judge heuristically, by simply looking at the interface and passing judgement according to ones own opinion. Hence, the outcome comprehends a list of personal statements about what is good and bad about the interface.

The main goal of heuristic evaluations is to identify any problems associated with the design of user interfaces. While a large group of evaluators can be expensive, it has been shown that 3-5 experts can already identify 75-80% of known usability flaws [182]. Moreover, practised evaluators can produce high quality results in a limited time. In [182], Nielsen presents ten ‘heuristics’, with maximum explanatory power, for user interface design:

- Visibility of system status: The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- Match between system and the real world: The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.
- User control and freedom: Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- Consistency and standards: Users should not have to wonder whether different words, situations, or actions mean the same thing.
- Error prevention: Eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
- Recognition rather than recall: Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another.
- Flexibility and efficiency of use: Allow users to tailor frequent actions.
- Aesthetic and minimalist design: Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- Help users recognize, diagnose, and recover from errors: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- Help and documentation: Supportive information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.

To perform a heuristic evaluation on the iFanzy interface, we asked the students to use the iFanzy Web interface intensively for one week. Afterwards, they had to evaluate their experiences and opinions given the ten heuristics composed by Nielsen.

Test results

Out of these three tests, performed by 60 students and many more test subjects selected by those students, a huge amount of feedback was obtained. However by summarizing this feedback, we could clearly identify the most occurring problems, bugs and weaknesses within this interface:
• Icons: People had difficulties with various icons we use. They were often too small, and not always clear enough in conveying what they stand for.

Discussion: As previously mentioned, on the iFanzy Web site we use icons to indicate actions a user can execute on a program’s details page (As shown at the bottom of Figure 9.6). These actions include “Add program to your personal selection” (the “sheet of paper” icon), “Add program to your favorites” (the “human heart” icon), “Record program” (the “round with capital ‘R’” icon), “Set a program reminder” (the “alarm clock” icon) and “Recommend program to a friend” (the ‘envelope’ icon). While two icons were clear (an alarm clock for a reminder and a human heart for the programs you love and favor), the other three were not considered intuitive enough. We agree that these three can be hard to understand. A potential reason might be that the associated actions are not commonly encountered on the Web and therefore users need to get used to them. From the other perspective, actions like “Add program to your personal selection” are just difficult to convey by means of one little icon. A bit more documentation, especially for new users, might be an option. A larger icon size has already been implemented.

• Rating: Subjects often reported problems with the rating bar. People found it hard to see the difference between ratings they entered and recommendations generated by the system. For example, they found it hard to understand why the same program could get a different system rating when it was broadcast on different channels or at different times. Some people indicated that they would like the possibility to rate a program independently of the current context.

Discussion: This issue clearly shows that the user does not understand the difference between a program in general and a broadcast of a program which includes context. E.g. I like movie X, but not so much when it is broadcast on a commercial channel which interrupts it every 15 minutes. Or, I like X but not at 8am. Because of this context, iFanzy can generate different scores for the same program when it is broadcast at different times and/or on different channels. Apparently, users do not fully comprehend this concept of context-sensitive recommendations yet. Therefore, we need to make sure that we explain our approach more clearly, for example by means of extra documentation and/or tooltips.

• Undo: Users indicated that they would have liked a button to undo a given rating.

Discussion: We fully agree that an undo-feature would be helpful for users when using iFanzy. The reason for lacking this feature lies in the fact that every rating is immediately sent to the server where it is treated as an event and its influence propagates throughout the user model. At the time of the test, undoing this series of events proved to be complex. However, currently we are working on facilities which allow the user to undo or adjust previously given ratings.

• Error messages: When an error occurs, iFanzy shows the standard error page without an explanation about what exactly went wrong.

Discussion: Indeed, sometimes errors occur after which the user is forwarded to a standard error page. However, coming to this page signifies a critical error from which the Web interface could not recover properly. The cause of such an error can originate from different layers within the application, and we do not want to bother the user with the technical aspects of such an error. Currently, a more advanced error handling system is being devised.

• Scaling: On devices with low resolutions, like an Internet-capable mobile phone, the interface fails in providing a good overview.
Discussion: Indeed, making the Web interface scalable on low resolution devices, was never a priority. This was mainly due to our focus on a dedicated iPhone application where the interface is specifically adapted to the platform.

- Tabs: Users found the tabs at the top of the page too small, too low in contrast or in general not conspicuous enough.

Discussion: This issue is probably due to us being too familiar with the interface and therefore not noticing that contrast was too low. Of course, it can also depend on the contrast settings of a monitor. In the mean time, this issue has been resolved by changing colors improving the contrast.

- Loading times: Often, users found the loading times of the PEPG too long, and sometimes thought the Web site had crashed.

Discussion: With our previous server implementation, on which this user test ran, the general performance in terms of speed and responsiveness was low, especially when multiple users were using it at the same time. However, with the latest server incarnation, all functionality has been rebuild with a strong focus on speed and scalability. By introducing better threading, spreading functionality over different physical servers and an optimized database layout (as discussed in Chapter 8), responsiveness has improved tremendously.

- (P)EPG time blocks: People did not like the fact that time blocks (of three hours) were fixed. E.g. with blocks from 18:00-21:00 and 21:00-00:00, it is impossible to see what plays between 20:00 and 22:00 in one screen.

Discussion: This issue uncovered a clear mistake in the interface design. In the latest online version this issue has been resolved by using flexible time blocks.

- Colors: In the iFanzy EPG, we give programs that fit the user a shade of orange where a stronger shade indicates a better match. However, when a program does not match the user at all, the evaluators missed a color to indicate how bad the program matched for them (e.g. in shades of red).

Discussion: Only showing colors for positive program matches was a clear choice. We believe that introducing more colors (e.g. shades indication how bad a match was) would also lead to a less orderly interface, potentially confusing the user even more. Moreover, we are convinced that we should emphasize positive matches as these are the programs the user would be inclined to watch. However, we might consider only giving the very worst matches a color, such that the user knows which programs do not fit at all. From a different perspective, it might also be possible that confusion was created due to our context-sensitive recommendation approach (see also the second bullet). It might for example occur that program X is colored (and thus recommended) in situation A and appear grey in situation B, which is possible if the recommendation score hangs around the threshold. Here again, extra documentation could provide helpful.

- Browsers: Some users indicated different behavior and/or different performance depending on which browser was used.

Discussion: iFanzy makes extensive use of Ajax to render functionality asynchronously. Moreover, we always try to follow the Ajax standard and best practises as much as possible. However, some browsers are known to render the same Ajax code in a different fashion. This is due to the fact that browsers tend to have their own custom JavaScript rendering engine. Although we do our best to make the Web site look as solid as possible on every browser, or at least the most common ones, it remains possible that small differences exist.
Besides these general issues, also many little bugs in design, logic and operation were identified. Even some grammatical and spelling mistakes as well as some erroneous links were reported and solved.

In general we can say that the three evaluation methods were successful in uncovering many problems, bugs and design mistakes. However, we also saw that some confusion originates from the fact that there is not enough documentation or explanations to inform people about how iFanzy works. After the evaluation and the processing of all feedback, the results were acknowledged and contributed to a new and updated version of the iFanzy Web interface, were most of the identified issues were solved.

9.5 Conclusion

In this chapter, we gave an overview of the client applications, running on top of the iFanzy server, allowing the user to benefit from the techniques that our approach enables. We introduced the server implementation and architecture of the approach presented in Chapter 4, explaining how we practically amass program metadata and make it available for personalization and recommendation algorithms.

Further, we introduced the SenSee interface which was the first client application running on top of this server. Although perfectly functional, the SenSee interface was never a real end product, ready to be used by the larger mass. SenSee served as a proof-of-concept to show the potential of the approach to a larger public, and for us as an evaluation environment to fine-tune operations. The real user interface breakthrough to the larger public came with iFanzy. In this chapter, we showcased the interfaces of the three currently available iFanzy client applications. Together, they shape the ubiquitous nature of iFanzy like envisioned in Chapter 2 and more specifically in Figure 2.6.

To figure out how the end-user experiences the iFanzy interface, we ran a user study on the iFanzy Web interface. By means of three methods, including a Cognitive Walkthrough, Thinking Aloud experiment and a Heuristic Evaluation, we could clearly identify both positive and negative points in the interface. Interestingly, most people complained about exact the same points in such a consistent manner that it gave us the perfect feedback to further update and improve the interface.

Recapitulating, in this chapter we addresses research question 8 (How can the user interact with such a system, following our proposed approach and applied in the television domain, effectively?), as a set of interfaces allowing a user to interact and benefit from our approach.

9.6 Related Work

In this chapter we discussed iFanzy, a platform targeted at providing a seamless and ubiquitous television experience to the user. However, due to the complexity of such a large project, researches often restrict themselves to investigate smaller scale aspects bridging the gap between the television platform and more common Internet services. For example in [235], Tullio et al. experiment with voice and text chat on the television, while in [67] Coppens et al. try to bring buddy lists, invite a friend function, a calendar, etc. to the television platform. In [60], Cesar et al. present an in-depth overview of the evolution of TV systems in terms of content and interactivity.

However, some researchers also present some larger scale projects more comparable to iFanzy as a whole. In [217], Smith et al. present the Personalized Television listings system (PTV). PTV tries to tackle the information overload problem associated with TV listings by providing an online personalized TV listings service. This service facilitates that each registered user receives a daily TV guide that has been specially compiled to suit his or her particular viewing preferences. This TV guide or Personalized Electronic Program Guide (PEPG), notifies the user about the programs which fit him best. Just like in iFanzy, this system requires a user model, which is obtained from rating (explicit) feedback provided by the user. However, no implicit information
is elicited. To overcome the cold start problem, they allow the user to provide preliminary profile information at registration time. Unlike iFanzy, they do not use any external source to make this process less obtrusive. To make these recommendations, they integrate user profiling, Case-Based Reasoning (CBR) and Collaborative Filtering (CF) techniques, an approach which is very similar to ours. However, the strength of iFanzy comes from the fact that we utilize Semantic Web techniques to support this personalization. While they use a simplified data model to describe the program metadata, we use a well-structured data model which is filled and enriched by utilizing knowledge from various sources. Further we employ a tight coupling between both data model and user model. After all, a richer data model is better in supporting algorithm like CBR and CF which depend on the calculated similarity between programs on one hand and users on the other. In [188], O’Sullivan et al. present PTVPlus, a commercialized extension on PTV. In this paper they focus on improving their CF strategy by the use of data mining techniques as a way of supplementing meagre ratings-based profile knowledge with additional item-similarity knowledge that can be automatically discovered by mining user profiles. Although iFanzy does not support CF yet, we have a similar approach when we use data mining techniques to discover viewing patterns from the historical viewing behavior of the Dutch population.

TiVo is one of the most successful commercial attempts to bring a more advanced Digital Video Recorder (DVR) to the market [21]. In 2008, TiVo served close to four million families in the US alone. The main feature that distinguished TiVo from a regular video recorder was the fact that TiVo had a build-in hard drive, recording video digitally. Further, its, in those days, really revolutionary feature was the ability to automatically record programs. It could record not only specific requests or all episodes of a particular season, but also content the user is likely to be interested in. This feature, called TiVo Suggestions, is based on explicit user ratings implemented by means of “thumbs up” and “thumbs down” buttons. Through these ratings, TiVo compares the behavior of different TiVo users to apply collaborative filtering to suggest also other programs. Through this functionality, TiVo made a first clear step from a regular DVR to a PVR or Personal Video Recorder. Comparing to iFanzy, TiVo focusses very much on a set-top box based system, while iFanzy delivers a more open and ubiquitous environment including Web and mobile access.

In [9] and [10], Ardissono et al. propose the multi-agent architecture of a system for the generation of adaptive EPGs, which filters the information about TV programs depending on the user’s interests, taking time context into account. The system consists of agents collecting data about programs, monitoring the user’s behavior to deduce interests and selecting advertisements, depending on preferences and context, to be shown in the PEPG. While this outline looks very similar to what we do in iFanzy, this system as a whole also appears to be limited. Firstly, the system solely runs on a set-top box environment excluding any cross-platform functionality. Secondly, although they use information from more sources (publisher Web sites and DVB metadata), the data integration is superficial. Only refinement of genres, via a custom hierarchical genre tree, is applied. In [10], the focus lies specifically on the user modeling and recommendation strategy. The User Modeling Component (UMC) they employ predicts interests in programs by looking at explicit, dynamic and stereotypical information. For every one of these three aspects, a confidence level is calculated indicating the respective uncertainty which is later applied as a weighting factor in the consolidation of a predicted interest. The recommendation strategy is a custom algorithm relying on these interests predicted by the UMC. Still, the data model on which they base these predictions is rather simple in comparison to our semantic data graph. This inevitably leads to the fact that other recommendation strategies, which depend on similarity calculations like case-based reasoning and collaborative filtering or any hybrid solution, would have difficulties to perform.

AVATAR is an example of a TV recommender system which makes use of reasoning with TV-Anytime semantics for TV content and user preferences to get valuable results [41]. In this approach the authors make use of a combination of TV-Anytime metadata fields and a custom-made genre hierarchy (which induces a possible compatibility issue with the official TV-Anytime topic hierarchies), together forming an ontology. Moreover, they do not reuse knowledge from any other ontology or vocabulary, besides TV-Anytime. Their user model consists of a list of values

\[^{4}\text{http://www.ptvplus.com/}\]
which expresses the interest in a certain concept of the ontology or instance of such a concept. Every such a value is a DOI or Degree Of Interest which is calculated based on both the structure of the ontology (e.g. the DOI of class siblings, depth of a genre in the tree, etc.) as well as on statistics like “percentage of the program watched by the user” or “time elapsed until the user decided to watch a recommended program”. However, a downside of this approach is that whenever a value is updated, this update propagates through the entire ontology and leads to many recalculations of connected DOI values. With a large data graph or a lot of instances, this is not a very scalable solution. Further, their recommendation algorithm is an effective hybrid combination of content-based and collaborative filtering. However, due to their complex calculation of the DOI value, also the calculation of the groups and inter-person similarities becomes a time-consuming task. Lastly, compared to iFanzy, they do not employ any form of contextualization of feedback and do not propose any solutions to the cold start problem.
Chapter 10

Evaluation

Evaluating a large application or framework is not an easy task. Often, large applications have architectures with many different dependencies and various interconnected components. The fact that so many applications nowadays consist of several different components, is a result of the evolution towards *Component-based software engineering* [119], which specifically emphasizes the concept of *separation of concerns*. By applying separation of concerns, an architecture becomes divided in such a way that every part or module deals with a distinct problem, with as few overlap as possible between modules.

To evaluate such an application, many approaches have been devised as shown in the survey presented in [149]. This survey includes approaches which evaluate every component individually, focus on the connections between components or evaluate the system as a whole. The latter is sometimes also referred to as an *end-to-end evaluation*, of which concrete examples among others can be found in [247] and [104]. End-to-end testing is similar to system testing; the ‘macro’ end of the test scale; involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate [153]. Therefore, such an evaluation is usually performed in order to verify the general coverage of the application, guide development efforts and/or to track improvements over time [104]. Moreover, when given the same inputs, an end-to-end evaluation enables a fair comparison of different versions or implementations of a system.

While Chapter 9 previously explained how iFanzy, following the approach defined in Chapter 4, works and looks, in this chapter we report on the user study we executed to measure the quality of the iFanzy recommendation strategy. With this user study we identify certain recommendation characteristics which were perceived by the users as important for determining the quality of a recommendation. Focussing at these characteristics, in this chapter we measure how well our strategy performs, and provide an answer to research questions 9A (*What characteristics of recommendations are perceived by users as important for determining the quality of a recommendation?*) and 9B (*How can we measure the quality of recommendations generated by our approach?*).

10.1 Recommendation evaluation

There are many facets of iFanzy which are eligible to be tested and evaluated. Client usability, user interfaces, server performance, quality of service, etc. to name just a few. Yet, the main crux of the system, discerning iFanzy from the competition, remains the quality of the recommendation engine supported by a rich and well-considered data model on the one hand and a profound and reliable user model on the other. In iFanzy, as presented in previous chapters, several different techniques and strategies are deployed and combined in order to generate these personalized recommendations. As explained at the start of this chapter, in such a situation, an end-to-end evaluation lends itself perfectly to evaluate the functionality of the system as a whole. After
all, we want to evaluate whether or not our complete approach (including semantic enrichments, the event-driven user model strategy, etc.) is able to facilitate data retrieval, under the form of recommendations, which can serve the end-user in his or her quest for information and provide satisfactory results.

Furthermore, an interesting side effect of the end-to-end evaluation is that it enables a fair comparison of different versions of the system. This is particularly handy for example compare a baseline version of the recommendation system with an extended version of the algorithm (e.g. including an extra feature). Such a comparison could then indicate the influence of that extra feature leading to the insight whether or not to include it further or to quantify its significance.

For the evaluation of the recommendation algorithm, a group of 60 people used the iFanzy Web interface for about two full weeks. This work was done in collaboration with two former master students at Stoneroos, who did their research within this project. They facilitated the technical implementation which made this evaluation possible, and for further details we would like to refer to their thesis [241]. In the following three subsections we discuss the evaluation’s goals, setting and results respectively. In the last two subsections we compare our results to similar work and further discuss our evaluation strategy.

### 10.1.1 Goals

The main goal of this user study is to see how ‘good’ our recommendation system performs. Unfortunately, a ‘good’ recommendation is difficult to define. Due to both psychological and technical reasons, measuring a recommender’s quality is not a straightforward task [123]. From a technical viewpoint, we can for example say that different recommendation algorithms perform better or worse given a specific data set or when running under certain restrictions. E.g. we know that the Collaborative Filtering algorithm only starts performing when there are far more users than items in the system. Even more challenging however, is the human factor. Different people behave differently, and we see for example that cultural background, social class, etc. influences recommendation satisfaction. For example in [232], Torres et al. show the similarities and differences between American and Brazilian people in a recommender test. Therefore, for algorithms it remains a difficult task to identify a ‘good’ recommendation in the same way a user would. Imagine for example that a television recommender recommends a set of programs which are reruns from programs which aired last week. The user, who already saw them last week, would probably not be very satisfied with such a recommendation. However, from a technical accuracy viewpoint, this recommendation makes perfect sense. After all, the program still fits the user’s profile very well. In the end, as mentioned in [123], the bottom-line measure for recommender system success should be the user’s overall satisfaction and user satisfaction does not always correlate with high recommender accuracy [265]. Some commercial systems selling items, are able to measure user satisfaction by for example the number of purchased products. Hence, an acceptable measure of satisfaction in the television domain could therefore be formulated as: “if the system recommends program A and the user actually watches it, we can conclude that the user is satisfied with this particular recommendation”. However, the ability to inspect viewing behavior is not straightforward, and depends, besides on technical hurdles, on many other factors like for example privacy, STB hardware, agreements with STB providers, user contracts, etc. While it definitely resides on our wish list, until now iFanzy was not yet able to inspect the set-top box viewing behavior of a large group of users in a real-live environment.

In this user study, we would like to quantify the recommender’s quality when employing two versions of the system. The first version involved the baseline recommendation algorithm which only takes explicit user rating events into account. For the second version, the input of the algorithm also includes all behavioral information represented by the implicit feedback events. By evaluating these two versions of the system, we want to quantify the additional benefit of implicit behavior on top of the baseline recommender system (purely looking at explicit feedback). In other words, does monitoring implicit feedback actually contribute to a better recommendation? Until now we always presumed that behavioral data is a good predictor. Observing it in this user test would corroborate our approach. From these goals we can extract two hypotheses:
10.1. RECOMMENDATION EVALUATION

Hypothesis 1

*The more explicit user feedback the system receives, the higher its prediction accuracy.*

This hypothesis relates to the structure of our user model explained in Chapter 6. There, we have shown that every rating of resource $X$ also helps to estimate the user’s liking of other resources attached to $X$, by means of semantically described properties in the data graph (proposed in Chapter 5). Therefore, every rating propagates throughout the model, increasing the algorithm’s ability to approximate the user’s interest in programs, and therefore increases the accuracy. Naturally, we do expect this increase to level off after some time.

Hypothesis 2

*Implicit user feedback further improves the recommender’s accuracy.*

Further in Chapter 6, we discussed the gathering, modeling and data mining strategy of implicit user behavior. We believe there is much information to be found in the user’s implicit behavior because it is provided by natural and unbiased use of the application. Therefore, we believe that the exploitation of implicit feedback can further improve the recommender’s accuracy.

10.1.2 Setting

To evaluate the recommender’s quality, we need a considerable number of users to use the system and generate a considerable number of feedback, both explicit and implicit. After all, a large set of user events simulates longer term usage and allows the recommender to generate more stable recommendations which we need in order to evaluate the quality.

For this evaluation, a group of 60 people used the iFanzy Web interface for about two full weeks. This group was selected to represent people with different backgrounds and educations, as well as an equal dispersion in terms of gender and age. At registration time, users had to provide their gender, age and education level which were used for the generation of cold start recommendations. In Figures 10.1, 10.2 and 10.3 we see the distribution of gender, age and education respectively. In every of these figures we show the distribution of persons in our study on the left, and, for reasons of comparison, the distribution of the entire Dutch population on the right. It can be noted that we had slightly more males than females, we did not take persons younger than sixteen years old into account, our 65+ group was a little bit under-represented and the total group was slightly more educated than the national average (in Figure 10.3, the legend shows the increasing level of education starting with the lowest being “primary education” (basisonderwijs) and ending with Master degrees or higher (WO/Doctoraal)). The statistics on the entire Dutch population were obtained via the Dutch research group “Stichting Kijk Onderzoek”\(^1\).

Looking at these statistics, we see that differences in distribution between our selected users and the general Dutch population exist. Though, the biggest variation comes from the education dimension, while gender and age distributions are comparable. However, in previous work done by the Stoneroos team, we already concluded that the education dimension is the least significant one of these three, for the prediction of television programs [264]. Further, including education as a user test variable would require several hundreds of users more, to make the evaluation results statistically significant.

For this user study, the participants were asked to use the iFanzy Web site for two full weeks nonstop with a preferred minimum average of about 10 to 20 minutes of usage per day (which was the average time employees at Stoneroos daily spend on browsing their program guides). Usage includes for example rating programs, setting reminders, adding programs to their favorites, indicating which programs they watched, etc. Each of these actions result in the generation of an event as explained in Chapter 6. To encourage this usage, a specifically tailored version of the standard iFanzy Web site was made, by which the administrator could post, on a daily basis, little tasks. These tasks included for example: “Rate a program”, “Add a program to your favorites”.

\(^1\)http://www.kijkonderzoek.nl/
“Set a reminder for a program you do not want to miss”, etc. For every one of these tasks, we provided a little how-to, making the users familiar with the interface. To prevent these tasks from influencing the results, users were not restricted or forced to execute tasks in any way. These tasks merely showcased the possibilities of the interface to prevent that users would miss out on interesting and event-generating functionalities. After the test, every user had to fill in a questionnaire containing:

- A set of programs: We presented a heterogeneous list of fifty, previously unseen, prime-time programs to each user, and asked them to provide us with the optimal ratings (in the interval $[0,10]$) expressing their interest in these programs. In other words, the exact number conveying how much the user likes a particular program. This list of fifty programs was put together trying to make an equal distribution over both the major Dutch channels as well
10.1 RECOMMENDATION EVALUATION

as over the most frequently occurring program genres. In this way we could rule out any potential program bias.

- A set of questions: We asked a number of questions querying users to deduce how satisfied they were with the overall iFanzy performance and the specific recommendation quality.

To stimulate ‘natural’ behavior, the users were not informed about how iFanzy works and what happens behind the scenes.

10.1.3 Results

Questionnaire Results

From the questions posed at the end of the questionnaire, we present the results from the two most significant ones. The first question asked the users how satisfied they were with the recommendation scores assigned to TV programs at the very beginning of the test. Because at this point no user feedback has been given yet, these scores were calculated by the cold start generator and are therefore solely based on the three user characteristics (age, gender and education). In Figure 10.4 we see the distribution of the answers on this question. Here, we see that 19% was almost never happy with the recommendations, while 11% was often pleased with the results. In between, 38% was sometimes happy, while the recommendations could regularly please 32% of the people. In Figure 10.5 we see the distribution of answers on the second question, asking how satisfied users were with the recommendation scores assigned to TV programs at the end of the test. After two weeks of usage, we see that users are much more satisfied with their recommendations. While only 2% was never and 32% was sometimes happy, 28% was regularly, 34% often and 4% almost always pleased with the generated recommendations.

In Figure 10.6, we see the percentages of people who changed their minds in the time between these two questions. On the vertical scale, we see the results from the first question (Figure 10.4) and on the horizontal scale the results from the second question (Figure 10.5). If we then investigate for example the row of people which were regularly happy with the recommendation results at the beginning of the test, we see that at the end of the test 24% of them remained regularly happy. On the positive side, 35% was often happy and even 6% of them were almost always pleased at the end of the test. On the negative side, 29% was only sometimes pleased and 6% almost never. Looking at the whole figure, we can say that the group of people who were happier at the end (green part) largely outweighs the group of people which were not (red part). In total 26.3% of the people were equally happy at the beginning, as at the end of the test (yellow diagonal). Unfortunately, 17.9% of the people gave a lower score to the recommendations at the end of the test in comparison to the beginning. However, the majority of the people (55.8%) remarked that they were more satisfied with the recommendation quality at the end of the test. Since this question directly asked the users how they experienced the recommender’s quality, we can say that the overall user satisfaction of the generated recommendations grew over the course of two weeks.

User test statistics

With 60 users using iFanzy for two weeks, many explicit and implicit user events were generated. In Figure 10.7 and Table 10.1 we see the distribution of the various types of events, ordered by number of occurrences. By far, the most occurring event is the ‘RateContent’ event (44%), fired whenever a user rates a program. Other often occurring events are the ‘AddSelection’ event (to add a program to your personal selection) (11%), ‘AddWatched’ event (tell us which programs you already watched) (10%), ‘SearchContent’ event (to search for programs) (9%), etc. Table 10.1 further shows how often those events on average occurred per user. From this table we see that on average people rated almost 38 TV programs. However, the standard deviation of the rating

2Considering the fact that we were unable to obtain real watching behavior (e.g. from a set-top box), the user was able to indicate which programs he watched the previous days, in an effort to simulate this information.
CHAPTER 10. EVALUATION

Figure 10.4: How people initially perceived the recommendation scores

Figure 10.5: How people perceived the recommendation scores at the end of the test

Figure 10.6: User satisfaction evolution
10.1. RECOMMENDATION EVALUATION

<table>
<thead>
<tr>
<th>Event type</th>
<th>Count</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>RateContent</td>
<td>2067</td>
<td>37.6</td>
</tr>
<tr>
<td>AddSelection</td>
<td>528</td>
<td>9.6</td>
</tr>
<tr>
<td>AddWatched</td>
<td>444</td>
<td>8.1</td>
</tr>
<tr>
<td>SearchContent</td>
<td>408</td>
<td>7.4</td>
</tr>
<tr>
<td>ChangeChannelOrder</td>
<td>329</td>
<td>6</td>
</tr>
<tr>
<td>AddFavorites</td>
<td>289</td>
<td>5.3</td>
</tr>
<tr>
<td>SetRecording</td>
<td>270</td>
<td>4.9</td>
</tr>
<tr>
<td>SetReminder</td>
<td>227</td>
<td>4.1</td>
</tr>
<tr>
<td>Other</td>
<td>82</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Table 10.1: Numerical event statistics

Figure 10.7: Event type distribution

The event average was with 25.9 rather high. This indicated that some users uttered only few events and/or others rated much more programs than 38. Therefore, in an effort to continue only with the more active part of the group, we removed all users who uttered less than 12 explicit program ratings (average minus the standard deviation). By doing so, seven users were excluded from any further involvement in any performance calculations, leaving 53 usable participants.

In Figure 10.8 we see how the users rated programs by means of the rating scale (from 0 to 10 in steps of 0.5). In the introduction of Chapter 6 we saw that, backed by the YouTube movie rating statistics, users tend to rate with the extremes of the scale, and particularly the positive end of the scale. This pattern can also be recognized in Figure 10.8, albeit less extreme. We can say that people use the positive end of the scale more frequently (60% of the ratings is \( \geq 6 \) and shown in red)\(^3\) than the negative one (40% of the ratings is \(< 6 \) and shown in blue). Also similarly, ‘0’ (“I completely dislike this program”) is used considerably more frequent (142 ratings) than each of the other negative scores (from [0,5-4,5]). With 231 ratings, ‘8’ is the most often granted score.

Evaluation of the baseline recommender system

To evaluate the recommender system, we measure the recommender’s accuracy to deduce the algorithm’s performance. Although several accuracy metrics exist, we resorted to the class of Pre-

\(^3\)‘6’ is the minimum score for iFanzy to regard a program as a positive recommendation
dictive accuracy metrics [123]. These metrics measure the difference between the recommender’s predicted rating and the user’s rating (the optimal target value). In other words, the closer the recommender approximates the user’s value, the better. According to [123] these metrics are particularly important for evaluating tasks in which the predicted rating will be displayed to the user, which is the case in iFanzy where we show the predicted value in the rating bar of every program. A well-known predictive accuracy metric is the Mean Absolute Error or MAE, which can be seen in Equation 10.1. It calculates the average absolute deviation between a predicted rating ($f_i$) and the user’s rating ($y_i$) for every program $i$ of $n$. For the predicted rating $f_i$, we use in this evaluation the recommendation score $R_S$, as introduced in Formula 7.2.2. The MAE metric has often been used to evaluate other recommender systems, like in [45], [121] and [10]. Since the MAE is a measure of the average error, the recommendation accuracy rises if the MAE goes down.

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |f_i - y_i| \tag{10.1}
\]

The recommender algorithm can generate a recommendation score $R_S$ for any program at any time. To validate a prediction for a program $P$, we need to know how the user really thinks about $P$. As previously mentioned, at the end of this user study, we asked the users to rate fifty programs (carefully selected to cover the whole program space). These ratings are considered to be the ‘ideal’ values ($y_i$ in the formula), for those fifty programs, to which we can compare the recommender’s prediction by means of the MAE metric. In other words, these fifty questionnaire ratings are the test set in the evaluation. The $R_S$ value can be calculated at any time and depends on both the quality of the user model and the richness of the semantic data graph. Hence, our assumption is that the more user feedback the system receives, the better the prediction becomes. In this test, users have been generating events for two full weeks, and every event has an influence on the user model. Therefore, all the events generated by the users during those two weeks helped to create the user model which contains the facts on which the recommender can base its prediction. In this sense, we can see this generated user model as the training set of the recommender, before it gets tested against the test set. Note, that the questionnaire set of fifty programs was completely disjoint from the programs broadcast in these two weeks of usage.

In the evaluation, every user-generated event was recorded together with a time stamp of execution. Therefore, we could, after the test was finished, replay the sequence of events the user uttered over these two weeks of iFanzy usage. By doing so, we could create a stepwise evolution of the recommendation score, by recalculating it every time a new event was parsed into the user model. Hence, for each of these fifty programs and all 53 remaining users in the study, we could
10.1. RECOMMENDATION EVALUATION

recalculate the average MAE every time a new event was fired. Or in other words, we calculate the average MAE for those fifty programs having a user model containing 1 rating, then with 2 ratings, 3, 4, 5, etc., continuing until the user model contains all the rating events that user generated during the test. Do note that we are testing the baseline recommendation system, which only considers explicit rating events, ignoring implicit feedback completely.

In Figure 10.9 we see the average MAE (in blue), when gradually more user ratings are uttered, and the corresponding trend line (in red). At the start of the graph, when no user ratings are uttered yet, we see a MAE value close to 2.50. This means that on average the error of the predicted value is 2.50 points off, in a scale from 0 to 10. Still, we are now at cold start time and thus predictions are only based on stereotypes and statistics deduced from the user’s demographics. As the number of explicit user ratings gradually rises we see that the error drops and evolves towards a value of 1.85. This means that after about 40 user ratings, on average the recommender is 1.85 points away from the user’s ideal value. After 10 ratings the MAE was close to 2.20, after 20 it approached 2.05, after 34 it became 1.93 and after 39 ratings it became 1.88. At 34 ratings the data still represents 60% of the user population (60% of the users at least uttered 34 ratings), while at 39 ratings it drops to 50% of the population. After this 50% mark, the population was considered too small and thus the statistical error too big to draw further meaningful conclusions.

In this test, after 34 ratings the MAE became 1.93, which is an improvement of 22.8% in comparison to the error of 2.50 measured at cold start time. To test whether this improvement is significant, we performed a paired, two-tailed Student’s t-test with a resulting p-value of $1.21659e^{-6}$. Since the p-value is smaller than the significance level of 0.01, we can say that the data was statistically significant at an $\alpha$-level of 0.01. Therefore, we can conclude that Hypothesis 1 (The more explicit user feedback the systems receives, the higher its prediction accuracy) holds, although it starts leveling out after roughly 40 ratings.

![Figure 10.9: Average MAE evolution](image)

Including implicit feedback

Previously, we discussed the baseline recommendation system which solely relies on explicit feedback. In other words, only explicit program rating events were added to the user model. In this subsection, we investigate the inclusion of implicit events, and its influence on the MAE metric.

Previously in Section 6.2.2, we explained the influence of an event on the user model. If a user watches, records, favors or searches for program $P$, we assumed that he or she must have some
interest in $P$. Further, we saw that an implicit event like for example “adding a program to your favorites” led to an update and/or creation of liking assertions. This process of creating and/or updating likings, following from implicit events, was described by Formula 6.2.1. Within this formula we defined two variables, $L_0$ and $L_i$, which represented the initial value chosen to give to a newly created liking and the increment value to update an already existing liking, respectively. However in Section 6.2.3, we did not discuss the ideal values for $L_0$ and $L_i$, which are necessary to materialize the effect of implicit events in the user model. To approximate the optimal values of these variables, we exploit the results obtained in this user study.

As previously mentioned, the fact that all user events, generated in this evaluation, were saved and provided with a time stamp, allows us to replay the user’s actions, control the effect of these actions on the user model and evaluate the results via the MAE metric. The procedure to determine $L_0$ and $L_i$ is as following: we ascribe particular values to both $L_0$ and $L_i$, we replay all user events resulting in a particular user model, we let the recommender system calculate scores for the fifty programs the user rated in the questionnaire and finally we check the average error between these scores and the user-provided ratings via the MAE metric. For this procedure we randomly selected 20 users from the 53 user test participants. Then, this whole process was repeated several times, each time with different values for $L_0$ and $L_i$. Optimal values for $L_0$ and $L_i$ are obtained when the MAE metric reaches a minimum. Since we assume that different events have a different influence, a different $L_i$ should be determined for every type of event. In this brute force approach we take the five most relevant (implicit) event types (AddSelection, AddWatched, AddFavorites, SetRecording and SetReminder) into account. The range of $L_0$ was determined to be in [6-9], representing the positive end of our rating scale [0-10], not including the maximum score. For $L_i$, we limited the increment to a value within the interval [0-2], limited to tenths of a point (0.0, 0.1, 0.2, ..., 1.9, 2.0).

![Figure 10.10: Optimal influence for the five most relevant events](image)

After execution of this brute force approach, we could extract a set of combinations leading to the lowest MAE. In about 90% of these combinations the selected value for $L_0$ was ‘7’. The results for $L_i$ can be seen in Figure 10.10. In this figure we see the ideal contribution of how heavy every event type should weigh in fine-tuning the liking of a TV program. From this figure it becomes clear that the implicit ‘AddFavorites’ event should influence the liking the strongest (with an increment value of 1.4) and thus indicates best how much a user likes a program. Second to that, the events ‘AddWatched’, ‘SetRecording’ and ‘SetReminder’ are roughly equally important with a contribution of around 0.7 of a point. The weakest is the ‘AddSelection’ event which on
average should contribute only 0.3 of a point. The main reason here might be that people add programs to their selection out of habit or for another family member (e.g. I add a soap to my selection because I know my wife likes to watch it).

Now that we have determined the values for the variables $L_0$ and $L_i$ of Formula 6.2.1, we can further investigate the influence of implicit events on the user model. The procedure here is similar to the previous evaluation (only including explicit events), by again taking the explicit rate events as milestones. Every time a new rating is uttered by the user and added to the user model, we recalculate the recommendation score for each of the 50 questionnaire programs, and compare these values to the user’s ideal value by means of the MAE metric. However, unlike the previous test, now we also include the effect of the implicit events (following Formula 6.2.1 and using the values defined above for $L_0$ and $L_i$), which the user uttered randomly between his personal program ratings. Because of this procedure, we can plot the outcome of this algorithm, for the 33 remaining test subjects, in the same graph as the previous evaluation. In Figure 10.9, we see the results from this second test represented by the green line.

Recapitulating, when we looked at explicit events, after 34 ratings the MAE dropped from 2.50 to 1.93, which was an improvement of 22.8%. However, when we combine both explicit and implicit event information, the MAE value (after 34 ratings) dropped from 2.50 to 1.86 which is an improvement of 25.6%. Further, we see that the average additional benefit of implicit events on top of explicit events is rather limited (3.2%). After 40 ratings, the MAE value dropped to 1.79. However, we think that this additional benefit will grow larger when people make longer use of the system. The reasoning here is that people at some point stop or at least moderate explicitly rating programs, when they become happy with the recommendations. In [242], Setten already stated that people only tend to give feedback on incorrect predictions. This means that, as time progresses, we will depend more and more on implicit events, since people will keep on putting program reminders, set favorites, record programs, etc. Testing the significance of these results, we performed a paired, two-tailed Student’s t-test with a resulting p-value of $4.2706 \times 10^{-6}$, and thus statistically significant at an $\alpha$-level of 0.01. As a result, we can conclude that also Hypothesis 2 (Implicit user feedback further improves the recommender’s accuracy) holds, with the remark that we expect that its benefit will increase when people use the system longer.

10.1.4 Comparison

In the previous subsection we evaluated our recommendation strategy by measuring its accuracy via the MAE metric. Our approach reached a MAE value of 1.86 when 34 ratings were uttered (representing 60% of the user population) and 1.80 when 39 ratings were uttered (were 50% was represented). In this subsection, we investigate how our results compare to other TV programs recommendation approaches.

In [123], Herlocker et al. describe that, based on previous work from them [206, 108] and others, most recent recommendation system’s accuracy reached a so-called “magic barrier” which they are unable to break. The MAE value of this magic barrier is around 0.73 in a five star scale, and is therefore comparable to an error of 1.46 in a ten point scale like used in iFanzy. This barrier is approached by various recommendation strategies which are optimized for their specific application and domain. They all show this similar measure of quality which has led to the believe that natural variability may prevent getting much more accurate. The most probable reason to account for this barrier appeared to be the fact that users provide inconsistent ratings when asked to rate the same movie at different times.

In the evaluation of MovieLens, a movie recommendation engine based on Collaborative Filtering, an average MAE of 0.72 (or 1.44 in our scale) was reached [108]. While this is considerably lower than our 1.80, we also notice that MovieLens recommender runs within a more controlled environment, making it easier to make correct recommendations. MovieLens works for example on top of a fixed and stable list of movies. While in iFanzy, we work with a continuously changing set of television programs. Secondly, the MovieLens movies are all annotated with a set of metadata up to a comparable quality standard. In iFanzy however, the metadata quality differs and depends on the external sources from which we integrate data. Hence, some programs can be annotated
richly, while others can be described poorly. Further, in iFanzy we consider user context, since the user’s interests differ in different situations. Since context is not considered in MovieLens recommendations, they have to deal with less situational parameters. Lastly, in MovieLens people were asked to rate movies, reflecting how they feel about that particular movie as a standalone item. In iFanzy, people are influenced by the whole picture. If iFanzy calculates an almost perfect recommendation score for program \( A \), it will be unlikely that the user will fine-tune or adapt it to make it perfect. Or if two programs \( A \) and \( B \) are broadcast at the same time, and the user likes they both with a small preference for \( B \), \( A \) might be rated lower because of his high interest in \( B \).

In [10], Ardissono et al. discuss recommendation techniques for personalized EPG’s. They generate TV program scores by considering the user’s interests in program categories and broadcast channels. Further, they classify users in stereotypes by means of socio-demographic data, general interests and lifestyles. These stereotypes were extracted by exploiting information about the interests and behavior of TV viewers collected in the Auditel (2003) and Eurisko (2002) studies about the Italian population. In their first recommendation evaluation, they generated program predictions by solely looking at the stereotypical information, from the stereotypes in which the test users were classified, and then comparing them to explicit user feedback on those programs. The resulting MAE value was 1.3 in a scale from 0 to 5, comparable with 2.6 in our scale. While this result is worse than ours (even worse than our MAE value of 2.5 at cold start time), their results improved dramatically in a second test. In this second test, they included the subject’s viewing behavior, which they could extract from the Auditel panel including 5.000 Italian families and a total number of 14.000 subjects. Once their system was provided with real-life behavioral data, the MAE value dropped to 0.3 in a scale from 0 to 5, comparable with 0.6 in our scale. From this test, it becomes clear how important watching behavior, which we do not have in iFanzy, can be for a TV recommendation system.

### 10.1.5 Discussion

#### Difference between rating and recommendation score

In the evaluation of the recommendation engine, as explained in previous subsections, we use the Mean Absolute Error to measure the accuracy of the algorithm. As shown in Equation 10.1, we calculate the absolute difference between the system-generated recommendation score \( R_S \) on the one hand and the user’s rating, which he or she provided in the questionnaire and serves as the optimal value, on the other. However, these two numbers do not represent the exact same concept. The recommendation score, represents the prediction of the system reflecting how well a given program, in its current context, fits the user. After all, the same program can get a different score when it is broadcast on a different channel and/or different time. The user rating is the score given to the program, reflecting how the user feels about that program, independent of the program’s broadcast context. Therefore, a small discrepancy exists between the two numbers we compare to determine the accuracy of the system. Consequently, the presented MAE results will be slightly worse because of it.

This situation was created because we did not want to make the questionnaire overly complicated. In the correct case, we had to ask the user: “How much do you like this program if it would be broadcast on channel \( X \) at 8pm?” However, we believe that such questions could confuse the user, and consequently, potentially lead to inconsistent or less accurate user ratings. Nonetheless, we think that the impact of this choice on the algorithm’s accuracy will be limited. The major quality of the system results from its ability to integrate TV data from different sources, which gives the recommender plenty of metadata attributes and values to base its prediction on, and the extra semantic relations which are introduced in the data graph connecting different programs, genres, series, etc. The influence of the program’s context like channel and time of broadcast will undoubtedly play a role, although not an all-determining one.
10.2. INCREASING SERENDIPITY

Influence of User Interface design

Presumably, also the iFanzy user interface influenced this recommendation evaluation. As explained in Section 9.4, the iFanzy user interfaces indicate how well a program fits the user model by giving it a distinct shade of orange. However, this color is only applied to programs which fit the user positively (with a $R_S$ score in the interval [6-10]), leaving programs which do not fit the user (in the interval [0-6]), standardly colored grey. By doing so, we unintentionally only trigger users to give feedback on positively recommended programs, and do not encourage them to give feedback on programs they do not like. Therefore, users probably only rate programs which they like but were not recommended or adjust the rating of recommended programs whenever they did not agree with the generated score. Consequently, the user feedback we do get are corrections to the generated scores, which eventually lead to the algorithm better ‘understanding’ what the user expects. However, this effect should not be considered too harmful and could be expected, since the literature already stated that people in general only tend to give feedback on incorrect predictions [242].

10.2 Increasing Serendipity

The goal of a recommender system is to suggest interesting content which hopefully satisfies the user. However from the test, an often heard complaint of the recommendation system was that most of the recommendations are found to be too obvious, and not leading to surprising discoveries. In other words, understanding why a person might find a program interesting, may increase our understanding of which programs should be recommended, and thereby hopefully increase the serendipity of the system.

The reasons why somebody might find a program interesting can be very different, e.g. because they watch the program every day, because it is a documentary on a specific channel, because it is their favorite segment (e.g. ‘Action’), because an actor they like participates, etc. As previously seen, to do so, our recommender system requires descriptive program information, like for example the channel, start/end time, segment, type of program, etc., which is captured in a well-structured model as described in Chapter 5. This model can then be used to generate suggestions of programs that approximate as closely as possible what the user actually likes. Moreover, in the previous section we have shown that our approach performs reasonably good in this perspective. However, such explicit models, containing information about TV programs, can never account for all the reasons for which somebody might be inclined to watch a program. Imagine for example a user generally interested in music and in particular violins. While according to his user model he is not interested in watching talk shows, the fact that “Vanessa Mae” (a famous violinist) appears as guest in the talk show will make the specific show highly interesting to him. However, such subtle relations cannot easily be found and maintained without a specialized external source. Recommender systems relying on explicit models capturing metadata about items, therefore tend to overspecialize in the sense that they suggest only the most obvious content which maximizes the similarity between the program and the current user’s model, which could lead to reduced diversity [222]. Therefore, many other potentially interesting items which are not “close enough” to the profile according to the program metadata, might be missed. This is known as the “serendipity problem” [133]. We believe that serendipity is not an absolute notion, but a relative one in the sense that we can only state that one recommender system is more serendipitous than another one. Taking a recommender system $A$ as point of reference which exploits program metadata to generate suggestions, we say that system $B$ has a higher degree of serendipity if it suggests a number of programs i) that are ‘new’ or ‘fresh’ in the sense that they were not suggested by $A$, ii) that are liked by the user and iii) that are characterized by a lower degree of similarity (taking into account explicit program metadata) with respect to the programs that the user has watched so far.

In this section, we present an approach which can increase the serendipity of our recommender system which relies on explicit program metadata, by integrating a notion of ‘association’ or
"semantic relatedness" which tries to account for some of the reasons why somebody might like a program that are not explicit in the model. Our intuition here is that a recommender system may decide that a certain program is a relevant recommendation even when the users themselves are not expecting it based on the explicit metadata. To show that this approach of integrating a more lenient measure of ‘association’ between programs works, we take our recommender system and integrate Explicit Semantic Analysis [99] as a way to capture the semantic relatedness between TV programs. The work presented in this section was carried out in close collaboration with Geert-Jan Houben and Philipp Cimiano.

10.2.1 Explicit Semantic Analysis

Explicit Semantic Analysis (ESA) [99] attempts to index or classify a given document \( d \) with respect to a set of explicitly given external categories. Gabrilovich and Markovitch have outlined the general theory behind ESA and in particular described its instantiation to the case of using Wikipedia articles as external categories.

In essence, Explicit Semantic Analysis takes as input a document \( d \) and maps it to a high-dimensional real-valued vector space. This vector space is spanned by a Wikipedia database \( \mathcal{W}_k = \{a_1, \ldots, a_n\} \) in language \( L_k \) such that each dimension corresponds to an article \( a_i \). This mapping is given by the following function: \( \Phi_k : D \rightarrow \mathbb{R}^{\mathcal{W}_k} \) with \( \Phi_k(d) := (as(d, a_1), \ldots, as(d, a_n)) \).

The function \( as \) expresses the association strength between \( d \) and the Wikipedia article \( a_i \). In the original ESA model, \( as \) is defined by the sum of \( tf.idf \) values of all words of \( d = \{w_1, \ldots, w_s\} \) in the article \( a_i \) multiplied by \( tf \) in \( d \): \( as(d, a_i) := \sum_{w_j \in d} tf_d(w_j)idf_a_i(w_j) \).

As output we thus get a vector representing the strength of association of a document \( d \) with respect to the articles in Wikipedia \( \mathcal{W}_k \). These vectors can then be used to assess the similarity between documents at a conceptual level (e.g., using the cosine similarity between the documents indexed with respect to the Wikipedia articles, i.e., the vectors yielded by the \( \Phi \)-function). In our approach we calculate the semantic relatedness between TV programs by calculating the ESA similarity between their synopses and (sub-) titles. Intuitively, by mapping title and synopsis of programs into the Wikipedia article space, we yield a number of ‘concepts’ (actually Wikipedia articles) that are associated to or evoked by this program. For instance, if “Vanessa Mae” (the famous violinist) is mentioned as a guest in the synopsis of a talk show program, this synopsis will be associated to the Wikipedia categories violin, music, Guadagnini, electric violin, etc.

Calculating the similarity in the space of associated Wikipedia articles will then reveal associations which are not obvious from the textual metadata descriptions of the program.

10.2.2 Integrating ESA into the iFanzy recommender system

The ESA mapping for a given program is calculated by mapping the textual description as given by the title, subtitle and synopsis into the Wikipedia space. Because iFanzy mainly encompasses Dutch TV programs, we used the dump of the Dutch Wikipedia for this purpose. Given a program \( P \) and its textual representation \( text(P) \) we store the ESA mapping \( \Phi(text(P)) \) as an additional (computed) metadata property of \( P \). This property will contribute to the weighted matching score between programs by calculating the cosine similarity between the ESA vectors of the different programs. As mentioned in Section 7.2.4, the recommender works by calculating a weighted sum over all relevant metadata properties which match for two programs. The weight of each metadata attribute has been determined empirically and fine-tuned through user studies and domain expert feedback. The determination for the weights is done by optimizing them to minimize the MAE of the recommender score compared to the real user ratings on a randomly created training set derived from previous studies. This is the way we have proceeded also to determine the weight of the ESA similarity on the overall score of the recommender. In summary, ESA is integrated just as another score which contributes to the overall weighted score of a program. In our experiments we will quantify the impact of this additional feature by comparing the version of the recommender system with and without the ESA similarity score.
10.2.3 Evaluation

In order to assess the impact of the ESA similarity score on the recommender system and its potential to improve the performance of the recommender system, in particular increasing its serendipity, we executed a second evaluation on the data gathered in the user test described in the previous section. There we saw that, the more programs the user rated, the more data the recommender can exploit to make a good prediction. A prediction was regarded as ‘good’ if it came close to the (ideal) user rating. Goodness is thus inverse proportional to the error made (quantified by Mean Absolute Error metric). However, a first empirical analysis of the behavior of the recommender system showed that the most straightforward program metadata, was exploited in most of the recommended programs. We analyzed in how many cases of the top 10 recommendations to users, a certain type of metadata was used, finding that information about the genre of the program was used in 94.4% of the cases, the channel in 93.5% as well as the group or series the program belongs to in 65.3% of the cases. This shows that the recommender is using the same set of metadata for most recommendations and explains its sometimes predictable behavior. Further analyzing the 10 programs that every user liked most in the questionnaire (according to their ratings on a scale between 0 and 10), we observe that on average less than half (4.96) of these are actually present in the top 10 recommendations of the algorithm. Thus, we are missing more than half of the programs that are highly interesting for the user, which shows that an approach based on explicit program metadata only, is clearly not enough. In the following, we show that by integrating ESA we are in fact able to increase the serendipity of the system in the sense that programs which would not be suggested by their metadata alone, but which are nevertheless interesting to the user, are proposed.

Cold Start Phase

To evaluate the influence of the ESA similarity on the recommender’s quality, we made three distinct versions of the algorithm. The first is the unaltered recommender, here called the baseline algorithm (B), as discussed in the previous section. This version of the algorithm only takes explicit events into account, so no implicit events and thus no user behavior is included, minimizing the number of variables. The second algorithm is the baseline integrating ESA (B+ESA) and the third represents the baseline extended with a simple similarity measure quantifying the word overlap between textual descriptions of programs (B+WO). Figure 10.11 shows the Mean Absolute Error (MAE) comparing the score of the recommender to the real user ratings (which were provided in the questionnaire at the end of the user test) for these three versions of the system. The blue line in the figure, which represents the baseline, is the same as shown in Figure 10.9. However, take into account that the X-axis here is not represented linearly anymore. The start of the curve is stretched to emphasize the influence at cold start time.

We observe that in the cold start phase, where not many user ratings are available, the error rate is substantially reduced by integrating ESA or the Word Overlap similarity measure. Between 7 and 35 user ratings provided, B+ESA and B+WO are able to reduce the MAE by 0.15 (from 2.18 to 2.03) points (6.9% relative error reduction).

While the contribution of the simple word overlap based similarity measure and the ESA similarity seem to be comparable in the cold start phase, we will clearly show below that, in addition to being able to effectively reduce the error rate in the cold start phase, the ESA similarity also leads to the fact that the recommender systems discovers more ‘new’ programs compared to the baseline system.

Serendipity

The Mean Absolute Error is a measure of (inverse) accuracy. However, it has been often argued that when evaluating recommender systems, not only the accuracy or mean absolute error should

---

4 We have introduced this word overlap-based measure to be able to assess the real benefit of a more semantic similarity measure such as ESA compared to a purely syntactic one based on word overlap.
be taken into account. In particular, in [123] Herlocker et al. have argued that it is important to measure the ability of the system to come up with new and surprising content. Therefore, one of our main goals is to show that the baseline recommender system ($B$) extended with the ESA-based semantic relatedness measure ($B+ESA$) shows a higher degree of serendipity compared to the baseline system. According to our definition, a system (here $B+ESA$ and $B+WO$) is more serendipitous compared to a reference system (here $B$) if: i) it delivers a number of programs not recommended by $B$, ii) these new programs are liked by the user and iii) the similarity between these new programs and the user profile is low according to the baseline system (so that they indeed would not appear in the baseline’s top 10 recommendations). We show below that each of these conditions is met by our extension. When quantifying in how far the conditions are met we always take into account the top 10 programs recommended by each recommender system, as we assume that these are the ones the users will be most inclined to, to examine.

- **New programs**: Both $B+ESA$ and $B+WO$ recommend a number of programs that are not recommended by the baseline system $B$. On average, as Table 10.2 (column “Avg. # new $P$”) shows, $B+ESA$ recommends 4.28 new programs while $B+WO$ only recommends 2.92 new programs on average. This shows that the extensions indeed are recommending fresh programs and that the $B+ESA$ is superior in this respect to the $B+WO$ extension.

- **Appreciation by the users**: We quantify also in how far the users actually appreciate the fresh programs, looking at the average user ratings (from the questionnaire) for these fresh programs proposed by the extensions ($B+ESA$ and $B+WO$) in comparison to the programs recommended by the baseline $B$. As Table 10.2 shows (columns “Avg # new $P$ rated ↑” and “Avg # new $P$ rated ↑↑”), of the fresh programs suggested by $B+ESA$, 70% of the ratings of the users are at least ‘good’ ($\uparrow$ in the table) and “very good” (↑↑) in 52% of the cases. For $B+WO$ we get comparable results (69% of fresh programs rated at least good and 48% rated very good) at a lower number of fresh programs suggested. $B+ESA$ is clearly the

---

5 good is $>5$, very good is $\geq 7$, rated on a scale from 0 to 10
superior approach with respect to the serendipity, so that we only consider this extension in our further discussion below. Overall, comparing the lists of top 10 recommended programs for B and B+ESA we even observe that the average ratings of the programs recommended by B+ESA is higher (6.68) compared to the average ratings of the programs recommended by the baseline (6.53). We performed a paired, one-tailed Student’s t-tests to check if the observed differences between B+ESA and B+WO with respect to the baseline are indeed statistically significant and not due to chance. Table 10.2 gives the $\alpha$ levels at which the differences were found to be significant in brackets. The t-tests have been in particular used to compare the average ratings for the top 10 programs for all the test persons in the experiments for the baseline and our extensions. The increase is due to the fact that the average user rating of the fresh programs (in the top 10 of B+ESA) is higher (5.75) compared to the ones in the top 10 recommendations of B that were replaced by these new programs (5.45).

- Low explicit relation to user profile: To show that the fresh programs recommended by B+ESA have a low degree of similarity to the user profile according to the explicit metadata, we compare their average score (the recommender score used to rank results) according to the baseline system with the average recommender score of the top 10 programs suggested by the baseline. The average recommender score of the fresh programs is quite low (6.16) compared to the average score of the top 10 TV programs suggested by the baseline system (7.40). This shows that the relation between these fresh programs and the user profile is indeed much lower compared to the ones that are suggested, so that these programs would hardly ever appear in the top 10 suggestions of the baseline recommender system. This is undesirable as they are more appreciated by the user than the actually recommended programs, as argued above.

<table>
<thead>
<tr>
<th>System</th>
<th>Avg. rating top 10</th>
<th>Avg # new $P$ rated $\uparrow$</th>
<th>Avg # new $P$ rated $\uparrow\uparrow$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>6.5</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B+ESA</td>
<td>6.7 (sign. at $\alpha = 0.05$)</td>
<td>4.28</td>
<td>3.00 (70%)</td>
</tr>
<tr>
<td>B+WO</td>
<td>6.6</td>
<td>2.92</td>
<td>2.04 (69%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>Avg. rating top 10</th>
<th>Avg # new $P$ rated $\uparrow$</th>
<th>Avg # new $P$ rated $\uparrow\uparrow$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>5.6</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>B+ESA</td>
<td>6.1 (sign. at $\alpha = 0.01$)</td>
<td>5.84</td>
<td>4.32 (74%)</td>
</tr>
<tr>
<td>B+WO</td>
<td>5.9 (sign. at $\alpha = 0.01$)</td>
<td>4.52</td>
<td>3.04 (67%)</td>
</tr>
</tbody>
</table>

Table 10.2: Average user rating and number of new programs for our B+ESA extension compared to B+WO and B. Statistical significance and corresponding significance levels is indicated in brackets.

**Full vs. Text-only Metadata**

Unfortunately, rich metadata is not always available for all programs. So, an interesting question is how our extensions perform compared to the baseline in case only textual data (title, subtitle and synopsis) is available. We simulated this situation by removing all other metadata for every program, and repeated our tests. Table 10.2 shows that in the case of using text-only metadata the increase in average user rating for B+ESA compared to B is even higher and still statistically significant at an $\alpha$-level of 0.01.
10.3 Conclusion

Measuring the quality of a recommendation strategy is notoriously difficult. Different people value different aspects in different situations. In this chapter, we identified that recommendation accuracy is an important characteristic, since it directly measures the difference between how a user feels about an item and how the algorithm approximates it. Further, an often heard complaint of recommendation systems is that most of the recommendations are found to be too obvious, and not leading to surprising discoveries. Therefore, the serendipity of a system, which represents this effect, can be regarded as another important characteristic. Although also other relevant characteristics can be identified, these two were particularly important for users in our system, which thereby answers research question 9A (What characteristics of recommendations are perceived by users as important for determining the quality of a recommendation?).

Further, we investigated how we could evaluate our recommendation strategy. Given the complexities of evaluating our component-based architecture, we employed an end-to-end evaluation which is specifically useful to evaluate a system as a whole. The evaluation itself featured a group of 60 people using an adapted version of the iFanzy Web interface for about two full weeks. During this time, users uttered a multitude of both explicit and implicit events which in turn led to the construction of a user model. Given this user model, our recommendation system could generate recommendation scores for television programs. At the end of the test, users were asked to rate a set of 50 programs to obtain the optimal values, expressing how much the user liked those programs. Having the user’s ideal values on the one hand and the recommendation scores on the other, the Mean Absolute Error metric was deployed to measure the accuracy of the algorithm.

The results were obtained after executing two different tests. The first test evaluated the baseline recommendation system, which basically only took explicit user ratings into account. From this test we could see that as the number of explicit user ratings gradually rose, the error dropped and evolved towards a value of 1.85, after about 40 user ratings and in a scale from 0 to 10. At cold start time the error of the predicted value was 2.50 points off. The second test took, next to explicit events, also implicit events into account which resulted from the user’s behavior (e.g. setting program reminders, favorites, recordings, etc.). In this second test we could see that the recommender’s accuracy dropped further to a MAE value of 1.79. From the collected data we could confirm hypotheses 1 (“The more explicit user feedback the systems receives, the higher its prediction accuracy.”) and 2 (“Implicit user feedback further improves the recommender’s accuracy.”).

Identifying the importance of the degree of serendipity of a recommender, we applied the ESA similarity technique in an effort to increase it. The ESA similarity framework attempts to index a given program with respect to a set of explicitly given external categories. By using all Wikipedia documents as categories, for every program we calculated a vector showing how well this program relates to any to those categories. Doing this for all programs, we could find inter-program relations which can not be found by only looking at explicit metadata. In this chapter, we have shown that based on the results of the previous user test, our approach indeed increases serendipity.

With the execution of this user study and measuring the quality of the recommendation strategy in terms of accuracy and serendipity, we provided an answer to research question 9B (How can we measure the quality of recommendations generated by our approach?).

10.4 Related Work

In this section we focus on other research looking at strategies increasing serendipity in recommendation and information retrieval systems. The topic of introducing serendipity into information retrieval, recommender and information filtering systems in general has received considerable attention in the last years. Toms discusses the importance and role of serendipity in information retrieval [230] and proposes several ways by which serendipity can be facilitated, among them: i) “blind luck” paradigm (making random suggestions about what the user might be interested in),
ii) “pasteur principle” (exploiting the user profile to discover information items that might be too similar to the current interests and preferences of the user to be surprising), or iii) “anomalies and exceptions” (exploiting ‘poor’ similarity measures to make unexpected recommendations).

The idea of integrating “poor similarity” measures has been for example exploited in recommender system research [133]. While Iaquinta et al. do not strictly follow the advice of incorporating a “poor similarity” measure, they approximate serendipity as “being half way” between maximal relevance and maximal irrelevance. While our approach follows a similar architecture as their approach, we have aimed at integrating some sort of “other similarity” (instead of poor) which tries to model associative behavior and thus come up with more serendipitous suggestions.

In general, the recognition of the fact that some degree of serendipity is important for recommender systems has led to the critical rethinking of current evaluation measures. McNee and colleagues [162] have argued that evaluations only based on accuracy (precision, recall and mean absolute error) are not only not sufficient to assess the usefulness of a recommender system but also detrimental in the sense that recommender systems have been tuned to come up with suggestions that are as close as possible to the users profile, thus being very predictable and hardly making any interesting suggestions. In this line, some researchers have made concrete suggestions for measures that evaluate the serendipity of recommender systems such as the unexpectedness measure proposed in [178]. We have also tried to quantify the unexpectedness of our recommendations by calculating their similarity to the user profile according to the explicit program metadata and shown that the $B+ESA$ approach indeed comes up with unexpected recommendations (in the sense that they have a much lower score that the recommendations of the top 10 items suggested by the baseline system $B$).
Chapter 11

Conclusions and Future Work

In this chapter we recapitulate the results from our research, and provide future research directions.

11.1 Conclusions

In this dissertation we have addressed several research questions dealing with the definition of an approach to support the user in finding information in a context-sensitive and personalized way. To demonstrate and evaluate this approach we have applied it in the television domain, eventually leading to the commercial launching of iFanzy, a personalized electronic program guide.

In this section we enumerate the different research questions postulated in Chapter 1, and summarize the conclusions which were drawn in the respective chapters of this dissertation, providing answers to those questions.

Research Question 1

Which technologies and standards, prevalent in the television domain, exist and can be suitable to support interactive and personalized television applications?

We answered this question in Chapter 2, by thoroughly introducing the television domain in all of its facets. We provided the state-of-the-art of digital television applications, existing at the start of this dissertation, and explained all of the technologies which are used to facilitate such applications. Among those technologies we found for example OpenTV and MHP which are both popular middleware solutions, running on set-top boxes, in which digital television applications are developed.

At the start of this project, we also identified a clear hardware backlog which severely limited the possibilities of applications running on such set-top boxes. The lack of local storage, a return channel, fast and abundant memory, etc. prevented the breakthrough of more advanced applications. However, with current hardware evolutions in combination with the risen interest in the television as the new home media platform, we can already see now that set-top box capabilities are growing quickly.

Further, we saw that the, by far, most useful and known digital TV application involves the Electronic Program Guide or EPG. An EPG basically shows an overview of programs playing now and in the near future, such that users have a complete summary of the available channels at their disposal. However, when, because of the digitalization, more and more channels, Video-On-Demand content, Web channels, user generated content, etc. become available, even an EPG will become insufficient to select the best program to watch. A user will have to choose a program from thousands of possibilities. This problem is larger than just the television domain, and occurs whenever the search space within a specific domain grows to huge dimensions.

To provide an answer to this recurring problem in general, we investigated the definition of an approach which can support the user in finding the best available data in any specific situation.
However to do so, we already anticipated that a good knowledge of the domain and a rich description of the relevant domain items is key. Therefore in that chapter, we gave an overview of the most important audiovisual metadata specifications, to describe television programs. Based on selection criteria like type, modeling and purpose of the specification, we selected five candidates with the potential to model relevant items in the television domain.

Research Question 2

What are the requirements for an approach providing user-adapted data retrieval?

In Chapter 3, we addressed research question 2. With the goal to define a generic approach to provide user-adapted data retrieval, we needed to formulate requirements of such an approach. Considering that already many domains are suffering from large amounts of information and the fact that many more will follow due to continuous data growth, our main requirement was to render this approach independent of the domain. After all, the most important requirements like for example rich item descriptions and a comprehensive user model are indispensable in any domain where personalization and intelligent data retrieval are leading functionalities. While pragmatically investigating the problem, we distilled three major areas of requirements necessary to reach user-adapted data retrieval: requirements with respect to the domain model, the user model and the adaptation process. Domain model requirements included for example the ability to describe a TV program class (and TV related classes) in all of its facets, reuse existing semantics as much as possible, link different related items, uniquely identify every resource, refer to audiovisual content like pictures, video, soundtracks, etc. as well as future resource types, describe a strong semantic structure, allow logical reasoning, express cardinalities, etc.

Thanks to the introduction of existing audiovisual metadata description schemes in Chapter 2 and the freshly defined data model requirements at the start of Chapter 3, we continued with the selection process of the most suitable metadata specification to support user-adapted data retrieval in the television domain. In Chapter 2, we already selected five potential candidates (including Escort, XML-TV, TV-Anytime, ProgramGuideML and LSCOM) which were eligible to provide the basis of the TV domain model. By comparing the newly defined requirements with the features of every metadata specification, we could conclude that TV-Anytime proved to be the best candidate. TV-Anytime can describe almost any possible program aspect including different distribution channels, program segments, related material, etc. Moreover, it makes extensive use of already existing schemes like MPEG-7 and Escort hierarchies. TV-Anytime includes a large number of classification hierarchies to classify programs from many different perspectives. Lastly, TV-Anytime is set up in such a way that it can easily allow for future extensions and/or links to other types of related content besides the regular pictures, video, music, etc.

For some requirements like for example the identifiability of resources, strong semantic structure and reasoning capabilities, TV-Anytime proved to be insufficient. However, these drawbacks were not so much due to TV-Anytime itself, but rather because XML was chosen as representational language. In an effort to solve these issues, we introduced the Semantic Web. By converting the XML version of TV-Anytime to RDFS/OWL, the widely acknowledged pivot languages of Semantic Web, we resolved the previously mentioned shortcomings. Further, having a Semantic Web representation of TV-Anytime introduced many more advantages like for example the interoperability between other existing RDFS/OWL domain descriptions.

Research Question 3

Which generic approach has the potential to provide user-adapted data access to large heterogeneous data sources?

The answer to this question was provided in Chapter 4, where we introduced the outline of our approach. Taking the requirements we formulated in Chapter 3, this approach was devised to provide user-adapted data access to large heterogeneous data sources independent of the target domain. The approach consists of three major components among which we find:
11.1. CONCLUSIONS

- The integration component: This component is responsible to gather and integrate, potentially heterogeneous, information, to be able to describe the relevant domain items as richly as possible. To do so, this component employs two types of external information sources. The first type encompasses domain-dependent sources which can increase the quality of existing properties or introduce additional ones. The second type involves domain-independent sources which provide more general background knowledge modeling for example domains like language, time, geography, etc.

- The user modeling component: With this component, we try to model the user by looking at all of his or her characteristics, preferences and interests relevant in the target domain. In our approach, this component maintains two different types of data, namely static and dynamic user data. Static user information encompasses all user data which remains relatively constant over longer periods of time, while dynamic user data encompasses all statements which are valid within a specific context. By interpreting the user’s context, we were able to determine very specifically the user’s interests at any specific point in time.

- The user-adapted data access component: This component focusses on the provisioning of user-adapted data retrieval. Given semantically annotated domain items, provided by the integration component, and one or more user model(s), maintained at the user modeling component, this component tries to adapt every step towards the personalized retrieval of information. It facilitates both personalized search as well as recommendations by taking the user’s socio-demographics, interests and context into account.

Each of these three components was discussed in details in Chapters 5, 6 and 7 respectively.

Research Question 4

*How can we integrate large heterogeneous data sources into one consistent and semantically rich data model?*

The answer to this question was provided in Chapter 5 which represents the integration component in the approach described in Chapter 4. After the conclusion in Chapter 3 that converting the TV-Anytime specification from XML into Semantic Web languages would be beneficial with respect to the requirements, Chapter 5 was started by discussing this conversion. Partially by relying on the ReDeFer approach, which describes among others the XSD2OWL tool, together with some manual translation of TV-Anytime’s data types, which could not be modeled in OWL, the TV-Anytime XSD specification was converted to an OWL representation.

Having an RDFS/OWL version of the TV-Anytime specification at our disposal, facilitated the integration of data from different external sources. As previously mentioned, in that chapter we described two types of data integration. To increase the expressivity of the TV-Anytime schema and the interoperability with other already existing schemata, we aligned TV-Anytime classes with external classes from other existing ontologies. However, we did not want to alter the newly created OWL representation of TV-Anytime, which would otherwise not be an exact OWL version of TV-Anytime anymore. Therefore, we subclassed the respective TV-Anytime classes, which we could then alter to make alignments to other external ontologies, thesauri, vocabularies, etc. This technique resulted in alignments to the OWL Time ontology describing time-related concepts, GeoNames and TGN describing geographical concepts, the WordNet thesaurus providing lexical relations and SKOS to introduce new relations between different concepts.

Besides introducing enrichments on the schema level, Chapter 5 also elucidated our integration approach to gather enrichments on the instance level. After all, items with a rich metadata description provide on the one hand more raw data which a reasoner can use to infer new relations and facts, and on the other hand more raw material to for example compare different items or compare items with a user model. Unfortunately, some sources publish their data with different semantics and/or different representational languages like XML or custom text formats. Therefore, it sometimes became necessary to transform those heterogeneous sources to fit our domain model and/or transform their data into RDF representations. After this transformation steps,
CHAPTER 11. CONCLUSIONS AND FUTURE WORK

all instance metadata adhered to our domain model structure. In this process we distinguished
root sources, which provide the broadcast timetables of various channels like e.g. BBC Backstage
and XML-TV, and enrichment sources which enrich the description of programs being broadcast
like for example DBpedia, IMDb, VideoDetective, etc. Again, data from these different sources
was integrated by using the OWL:sameAs construct and following the four basic Linked Open Data
rules outlined by Tim Berners-Lee.

Research Question 5

A: How can we model relevant user data to support context-sensitive adaptation?
B: How can we obtain this user data encompassing both explicit and implicit data?
C: How can we support new users who suffer from an empty user model?

The answer to these three questions was provided in Chapter 6, by introducing our user mod-
ing approach, as envisioned in Chapter 4. Previously in Chapter 3, we identified some necessary
user model requirements. Among those, one of the most important requirements was that we need
to be able to gather user data which is provided both explicitly as well as implicitly. After all, im-

citplicit feedback, which reflects the user’s behavior, is important to amass user data unobtrusively.
Furthermore, to get a grasp of the user’s interests at any point of time, we required user model
information to be contextualized.

Taking these requirements, Chapter 6 introduced the structure of our user model. On the one
hand, this user model maintained static information like specific user characteristics and prefer-
ences, which are unlikely to change much over time, while on the other hand also dynamic infor-
mation can be stored. To retrieve dynamic information we rely on an event-driven architecture.
Every action, both explicit and implicit, performed by the user on any of the clients connected to
the server, results in a user-generated event. These events are collected in the event model where
they can be evaluated before being materialized in the effective user model. Most explicit and
some implicit events, like for example the event generated when the user rates an item, are added
directly to the user model because they convey the user’s interest. Other implicit events are stored
within the event model, to be evaluated later by data mining algorithms to extract reoccurring
patterns. Patterns which are clearly present in the data can be materialized in the user model.
In Chapter 6 we deployed three data mining algorithms including JRip, NaiveBayes and Apriori.
These algorithms were used to learn user-specific rules or patterns from generated events, which
then in turn were added to the user model if the confidence in the rule was above a threshold.

Systems which rely heavily on user information in order to provide their key functionality,
usually suffer from the so-called cold start problem. Or in other words, the system tries to adapt
functionality to the user, but no user information is available to base crucial decisions on. In
Chapter 6 we illustrated three different strategies devised to alleviate this problem, by deducing
user information taking only a minimum of user characteristics (gender, age and education):

• Via historical viewing statistics: By obtaining general viewing statistics from the Dutch
market (including 33,000 program entries), together with the percentages of people, grouped
by age, gender and education, watching those programs, we were able to detect viewing
patterns indicating the average taste of users, by means of data mining algorithms. Such
patterns showed us which channels, genres and programs were most often watched by for
example a highly educated male person of 55 years old.

• Via import: By importing existing user data, coming for example from already existing
community profiles, we were able to get an idea of the user’s taste. Users publish so much
information on their online profiles, that often matches can be found with television related
items which in turn allow us to exploit that information. To prove this idea we parsed infor-
mation from the Dutch Hyves network, the largest community Web site of the Netherlands.

• Via classification: By classifying the user in a group, from which already some information
is known, properties of the group apply to some extent to the user himself as well. We used
group information from MovieLens and IMDb, to extract potential user interests.
Together, these three methods provide a reasonable amount of user information which is used to supply the user with a personalized experience, even when he or she is new to the system.

**Research Question 6**

*How can we provide user-adapted data access given a well-defined domain model and a comprehensive user model?*

The answer to this question was provided in Chapter 7, and elucidates the adaptation strategy in the approach described in Chapter 4. In Chapter 7 we introduced the concept of adaptive or personalized systems, which consider the user model in an effort to alter certain aspects of the system to better suit that particular user. However, besides the fact that adaptation can manifest itself in potentially every aspect of user data interaction, in Chapter 7 we focus on the adaptation of data retrieval in an effort to support the user in choosing the right items to consume.

The adaptation strategy which we presented in Chapter 7, is a stepwise process starting with a user request and ending with returning a set of results, which are specifically tailored to suit the current user. In Chapter 7, this process is depicted in the form of an “adaptation loop”, consisting of three components each contributing to the adaptation process:

- **Query refinement:** This component facilitates the first step in the adaptation process and is responsible to extend and enrich the original user request. To do so, we employed a number of sequential processes which first remove all stop words by means of the Lucene stop word filter, add the stems of words by means of the Porter stemmer and finally conceptualize keywords by searching for ontological resources semantically describing that keyword. However, since the conceptualization process can find several different matches for any given keyword, this process is succeeded by a concept selection step which tries to select the best match based on the user model (we only consider resources not disliked by the user) and a concept disambiguation process.

- **Querying and inferencing:** After having obtained an updated set of concepts and keywords that conveys the user’s request, a set of queries are sent to the knowledge base. However, since we do not only want to retrieve exact matches, this component tries to discover highly related and/or other potentially interesting programs by means of a reasoning engine. E.g. if the user searches for programs with the genre ‘Sport’ and the reasoner ‘knows’ that the genre ‘Sport news’ is related to the genre ‘Sport’, programs with the genre ‘Sport news’ will be included in the final result set as well (taking that the user model does not mention that the user dislikes this genre).

- **Content filtering:** The last step in the adaptation strategy involves the content filtering component. This component is able to, given a set of IF-THEN rules, further tweak the result set. This is for example necessary to filter out inappropriate results, rank results based on popularity (e.g. Video-On-Demand providers often want to promote their latest programs), filter resources (e.g. remove high definition content when the target device is a lower resolution smart phone), etc.

As a part of user-adapted data retrieval, in Chapter 7 we presented our recommendation algorithm to provide context-sensitive recommendations. The ultimate goal of any recommendation system is to make sure that a user has no need to search anymore, as the systems exactly knows when and which items to propose. The presented algorithm was devised as a hybrid prediction approach combining Demographics Filtering (DF) and Case-Based Reasoning (CBR). Demographics or stereotypes filtering relies on a set of stereotypes modeling a group of people, who behave similarly in terms of key characteristics important for the target domain. Knowledge of the stereotypes, as well as knowing to which stereotype a given user belongs, helps to determine what to recommend to this user. CBR on the other hand, tries to predict how a user feels about a new item, by looking at how items, rated in the past, relate to this new item.
CHAPTER 11. CONCLUSIONS AND FUTURE WORK

Research Question 7

Given our approach towards user-adapted data retrieval, how can we optimize data retrieval efficiency in terms of querying speed?

The answer to this question was provided in Chapter 8. The approach presented in Chapter 4 led to the ability to provide context-sensitive and user-adapted data retrieval to large heterogeneous data sources. However, systems dealing with large amounts of data and providing complex functionality like reasoning, user-adapted search, integration of data, recommendations, etc., require extra care with respect to querying speeds. Moreover, efficiency in terms of querying speed is vital for any system’s long-term success. Therefore, in Chapter 8, we introduced a number of optimizations to improve the efficiency of the database in terms of size and querying speed.

To be able to evaluate the benefit of each of those optimizations, we created a baseline implementation of the database system where no optimizations were applied. This ‘naive’ approach involved one single data repository, containing both all schematic information as well as all data instances. However, such an approach, from which we know that it will perform badly in our case, is often successfully applied in smaller-scaled projects. To evaluate the database system, we formulated a number of representative queries (in the TV domain) with increasing complexity, which we executed on this large data repository. The resulting querying times could then be compared later with the results of the execution of the same queries on a repository including specific optimizations.

Later in Chapter 8, we described several optimization techniques. The first important optimization involved the decomposition of the data set. Drawing from work in relational databases, such decompositions can be performed in two ways. Vertical, property-based decomposition is based on the schema; instances related to certain classes and properties are split off from the data set. Horizontal, instance-based decomposition is based on the resources; a set of instances with the same schematic structure is split off from the main data set. In that chapter, we applied vertical decomposition by splitting off data from IMDb and DBpedia, and horizontal decomposition by further dividing the IMDb data set.

Besides decomposition of the data set, we also performed other optimization techniques like for example using relational databases for well-structured parts of the data graph, the use of keyword indices and the utilization of the LIMIT clause in the query to restrict the number of results to be returned. To show the final cumulative improvement of all proposed optimizations together, we reran the four queries initially selected at the beginning of the chapter. The results of these queries executed on the fully optimized repository in comparison to the first naive approach of one large repository, were significant, without making any functional concessions. Querying speeds improved by a factor of 20 and in some cases even a factor of 130.

Research Question 8

How can the user interact with such a system, following our proposed approach and applied in the television domain, effectively?

The answer to this question was provided in Chapter 9, by proposing the SenSee and iFanzy user interfaces. To facilitate user interaction with these interfaces, we first introduced the implementation of a back-end server directly following the approach described in Chapter 4. The interfaces themselves therefore run as client applications on top of this server. The server’s architecture included a content retrieval layer for the retrieval and integration of metadata from different sources as described in Chapter 5, various services among which a user model service responsible for maintaining the user model as described in Chapter 6 and a personalization layer to provide user-adapted data access as described in Chapter 7.

SenSee was introduced as an illustrative Web application built within the project to allow context-sensitive interaction with the back-end server. This Web application uses state-of-the-art technologies like Google Web toolkit and Ajax to present its functionalities responsively. Within the interface, the user can search for television content using a standard search bar in combination with three facets in which respectively time frames, genres and locations can be selected.
Further in Chapter 9, we introduced iFanzy, a platform consisting of several commercialized client applications providing a personalized electronic program guide. The iFanzy client applications, including a Web portal, a set-top box interface and an iPhone application, run on the same server introduced earlier. Each of these client applications, is specifically tailored to meet the user’s expectations from each respective platform. While the iFanzy Web site for example provides a general overview of all available television channels, the iPhone application focusses on providing a quick overview of the programs playing tonight. Since all three applications, connect to the same server, every user action on any of them has an effect on all. E.g. rating a program via the iFanzy Web site, is immediately taken into account when a recommendation is generated on the set-top box interface. This cross-platform integration is what makes iFanzy a truly ubiquitous and immersive television experience.

Lastly, Chapter 9 concluded with a user interface evaluation of the iFanzy Web interface. By means of three methods, including a Cognitive Walkthrough, Thinking Aloud experiment and a Heuristic Evaluation, a group of 60 students and many more test subjects provided feedback on the design, implementation and logic of the Web interface. From these tests we could extract some small mistakes and bugs as well as some larger flaws in the design and operation. However, in the latest incarnation of the Web interface we were able to resolve most of the issues brought up, improving the user experience.

**Research Question 9**

A: *What characteristics of recommendations are perceived by users as important for determining the quality of a recommendation?*

B: *How can we measure the quality of recommendations generated by our approach?*

The answer to these two questions was provided in Chapter 10. In that chapter, we saw that evaluating recommendations is a difficult task, due to both psychological and technical reasons. Psychologically, because it is hard to define a ‘good’ recommendation and ‘good’ remains a very subjective notion. Consequently, for algorithms it remains an even harder task to identify a ‘good’ recommendation in the same way a user would. From a technical perspective, a ‘recommendation’ refers to a complex task depending on several different actors, subtasks and quality of relevant data sets which all influence the recommender’s quality, making it hard to evaluate it.

In this chapter, we identified that recommendation accuracy is an important characteristic, since it directly measures the difference between how a user feels about an item and how the algorithm approximates it. Further, an often heard complaint of recommendation systems is that most of the recommendations are found to be too obvious, and not leading to surprising discoveries. Therefore, the serendipity of a system, which represents the effect of surprising recommendations, can be regarded as another important characteristic. Although also other relevant characteristics can be identified, these two were particularly important for users in our system.

To evaluate our recommendation strategy, in Chapter 10 we deployed a so-called end-to-end evaluation which basically means that we considered the evaluation of the recommendation system as a whole more valuable than the evaluation of the recommender’s features individually. The evaluation itself featured a group of 60 people using the iFanzy Web interface for about two full weeks. To evaluate the recommendation’s quality itself, we measured the recommender’s accuracy to deduce the algorithm’s performance. Although several accuracy metrics exist, we resorted to the class of predictive accuracy metrics, and more specifically the Mean Absolute Error or MAE, also deployed in similar research. This metric basically measures the difference between the recommender’s predicted rating and the user’s rating (the optimal target value, obtained through a questionnaire at the end of the user test where users had to rate 40 arbitrary programs).

In Chapter 10 we presented the results from this evaluation, which consisted of two separate tests. The first test evaluated the baseline recommendation system, i.e. the recommendation algorithm which only considers explicit program ratings. In the second test, the recommender system took all user feedback, both explicit and implicit, into account. By separating the evaluation in these two tests, we could afterwards compare the exact influence of the inclusion of behavioral (or implicit) user data in the recommendation strategy.
With regards to serendipity, we implemented Explicit Semantic Analysis (ESA) as a technique to try to increase the degree of serendipity in the recommendation system. According to our definition, a system \( A \) is more serendipitous compared to a reference system \( B \) if: i) it delivers a number of programs not recommended by \( B \), ii) these new programs are liked by the user and iii) the similarity between these new programs and the user profile is low according to the baseline system (so that they indeed would not appear in the baseline’s top 10 recommendations). In Chapter 10, we showed that when our recommendation algorithm including the ESA extension, every of these requirements was met and generated more serendipitous recommendations.

### 11.2 Future Work

In this section we provide some pointers for possible future extensions with respect to the three core research questions underpinning our approach towards context-sensitive and user-adapted data retrieval. We conclude this section with work supporting the future of the television platform.

#### 11.2.1 Integration

As proved by the Linked Open Data Community, new sources are constantly being converted to RDF(S)/OWL, making the resulting data graph both semantically richer and covering more and more domains. Therefore, a straightforward extension involves including other relevant sources which can potentially help in constructing even richer TV program descriptions. Further, with more and more sources becoming available through SPARQL endpoints, we should give preference to utilizing these endpoints instead of keeping too much data locally.

Recently, OWL 2 Web Ontology Language [125], an ontology language for the Semantic Web with formally defined meaning, became a W3C recommendation. OWL 2 is the successor of OWL, fully backwards compatible and still based on an RDF serialization (although other syntactic formats are available). Almost all the building blocks of OWL 2 were present in OWL 1, albeit possibly under a different name. The most useful new feature of OWL 2, with respect to our work, involves the ability to define richer data types and data ranges. OWL 2 allows for example to define a property ‘adultAge’ which has a ‘minInclusive’ restriction of ‘18’, meaning that every ‘adultAge’ must be higher than or equal to 18 years old. Currently, we have to facilitate such restrictions via a rule engine activated after the content retrieval process.

#### 11.2.2 User modeling

In Chapter 6, we facilitated the extraction of user data from online social networks. However, gathering information from different user model repositories is a long and difficult process as it requires knowledge from the responsible source and the permission of the user to parse its contents. More interesting would be to investigate whether we can define a universal user model which serves as the central repository, including all known information about that user, available to all applications. Every single application would then utilize and update its contents leading to one consistent representation of the user. Moreover, the user can more easily maintain this repository and avoid for example compromising information to become available online.

In adaptive applications, scrutability is an often neglected subject. However, allowing the users to take a (restricted) look behind the scenes or giving them the ability to examine or update their user profile, can probably contribute in the user’s trust in the system. For example, explaining to the user how and why recommendations are generated could increase the user’s understanding and appreciation of generated recommendations. Moreover, it could for example allow users to improve their recommendations by updating some naive ratings they previously accidentally uttered. We believe that more scrutability can induce more satisfied users, and therefore could prove a valuable future extension.
11.2. FUTURE WORK

11.2.3 User-Adapted Data Access

The most visible part of our user-adapted data access strategy, is without a doubt the recommendation engine. A user is often confronted with its output and can therefore also quickly decide whether or not he or she likes the result. However, the recommendation strategy is also very susceptible to extensions or different approaches. While it would be easy to say that future work on the recommendation engine could for example include incorporating additional algorithms like Collaborative Filtering or devising better hybrid solutions, we would like to focus here on future extensions improving the whole recommendation experience.

A first future extension could involve to move a recommendation from simply generating a list of programs ordered by how good they match the user model, to a more ‘global’ recommendation which takes the user’s experience and other ‘surrounding’ programs into account. Imagine for example two programs being broadcast at the same time. Program A is a daily magazine which the user faithfully watches every day, and because of this behavior received a high recommender score. Program B is a movie which the user would enjoy a lot and therefore also received a high recommendation score. In the current system both programs would be highly recommended. However, a better approach would be to lower the score of program A, since program B is broadcast only rarely. In other words, the recommender would better consider the choice the user has to make (e.g. between channels and programs playing at the same time) instead of solely looking at every program individually. After all, in our current approach the user still needs to make a choice from the list of generated recommendations.

A second future extension could elaborate further on related research previously mentioned in this dissertation. In Chapter 6, we shortly touched the move.me prototype which illustrated a scenario for social interaction in which users can manipulate audiovisual sources through interaction with a sensor-enhanced pillow. This pillow contained sensors to measure how much a user was sweating (potentially indicating fear or anxiety), the pillow’s acceleration, etc. Potentially, it could also be extended with sensors to measure the user’s heartbeat, stress levels, etc. Although the pillow did not always function perfectly and sometimes failed in getting correct sensor readings, this approach holds some potential. Imagine that we do have the ability to perfectly measure the user’s physical and/or mental state, how could that information influence the audiovisual experience? We could for example show a warning, or change the channel in case a user really becomes frightened. Or even further, in case we would have excellent metadata describing every program segment (which is already supported in the TV-Anytime specification), a system could for example dynamically alter or remove scenes from a program on which the user reacts (too) heavily.

11.2.4 The future of television

For the majority of the people, TV media consumption occurs in the living room in front of a television set, and in some cases accompanied by a STB if digital services are available. However, we foresee that at some point the TV will become a part of the connected home where audiovisual content can be streamed to any other device, including for example personal computers, smart phones, tablet computers, etc. We also see an evolution where the STB will slowly disappear, having its functionality integrated in the television or the network itself. In such an environment, different devices can display different but complementary streams. Imagine for example watching a soccer match on the big screen, while the highlights are replayed on your smart phone. Further, also the concept of a “TV program” will most likely evolve from a fixed length broadcast into a personally rendered composition of dynamic fragments. For example when playing an action movie, bad language and/or violent scenes could dynamically and unobtrusively be removed from the program when children are around. Similarly, instead of watching the standard news broadcast, one could favor a quick personalized overview of the most important news items. Here, only the news which is relevant and/or favored (e.g. only foreign news, business news, etc.) would be included in the stream. Future work would therefore include the investigation of all facets of such a connected home, and how audiovisual content can be personalized and dynamically be reordered to stream to different devices.
Further, we expect the TV to evolve from a dumb display device into a personalized buddy, which can help people in their everyday life. We imagine for example how educational systems can be deployed to stimulate the educational process in children in a fun and pragmatic way. The system could for example keep track of the children’s progress, and automatically but subtly increase the difficulty level to promote advancement. Besides children, we also see a large benefit for the support of the elderly. Older people often suffer from loneliness and lack of social interaction, especially when they lose their mobility. There, the TV buddy could provide a portal to the world, letting them connect to friends and family, have video chats, play an online card game with others, etc. Even further, such a system could also for example monitor these people’s health, raising the alarm when something goes wrong.

One of the biggest annoyances when interacting with a TV system, are the remote controls. Nowadays, people have a different remote for the TV, for the STB, the PVR, etc. and get often confused which to use when. Moreover, with the evolution of TV capabilities, the remote control is becoming completely inadequate. Typing a URL, twittering some lines of text, editing your profile, etc., are emerging functionalities which require more advanced input than what a remote control currently can offer. However in the future, we see the possible deployment of smart phones to replace the remote. Such devices, like for example the iPhone, are perfectly equipped for typing texts or to facilitate more advanced interactions. Via IP, the phone can communicate with the connected TV set, to convey the user’s input. Even further in the future, we see great possibilities in controlling electronic devices by means of just thoughts. In ongoing research, scientists already achieved an active brain interface to let a monkey reach some fruits by means of an advanced robotic arm, just by thinking about it\(^1\).

\(^1\)[http://www.physorg.com/news194796581.html]
Appendix A

Data sources

A.1 XML-TV

A.1.1 XML-TV DTD

```xml
<!ELEMENT tv (channel*, programme*)>
<!ATTLIST tv date CDATA #IMPLIED
source-info-url CDATA #IMPLIED
source-info-name CDATA #IMPLIED
source-data-url CDATA #IMPLIED
generator-info-name CDATA #IMPLIED
generator-info-url CDATA #IMPLIED >

<!ELEMENT channel (display-name+, icon*, url*)>
<!ATTLIST channel id CDATA #REQUIRED

<!ELEMENT display-name (#PCDATA)>
<!ATTLIST display-name lang CDATA #IMPLIED

<!ELEMENT url (#PCDATA)>

<!ELEMENT programme (title*, sub-title*, desc*, credits?, date?,
category*, language?, orig-language?, url*,
length?, icon*, country*, episode-num?,
video?, audio?, previously-shown?,
premiere?, last-chance?, new?, subtitles*,
rating*, star-rating? )>
<!ATTLIST programme start CDATA #REQUIRED
stop CDATA #IMPLIED
pdc-start CDATA #IMPLIED
vps-start CDATA #IMPLIED
showview CDATA #IMPLIED
videoplus CDATA #IMPLIED
channel CDATA #REQUIRED
clumpidx CDATA "0/1" >

<!ELEMENT title (#PCDATA)>
<!ATTLIST title lang CDATA #IMPLIED

<!ELEMENT sub-title (#PCDATA)>
```
<!ELEMENT channel CDATA #IMPLIED>

<!ELEMENT premiere (#PCDATA)>
<!ATTLIST premiere lang CDATA #IMPLIED>

<!ELEMENT last-chance (#PCDATA)>
<!ATTLIST last-chance lang CDATA #IMPLIED>

<!ELEMENT new EMPTY>

<!ELEMENT subtitles (language?)>
<!ATTLIST subtitles type (teletext | onscreen) #IMPLIED>

<!ELEMENT rating (value, icon*)>
<!ATTLIST rating system CDATA #IMPLIED>

<!ELEMENT star-rating (value, icon*)>
A.1.2 XML-TV RDFS Schema
A.2 IMDb

A.2.1 IMDb Title RDFS Schema
A.2.2 IMDb Person RDFS Schema
A.2.3 IMDb Literature RDFS Schema

A.2.4 IMDb Company RDFS Schema

A.2.5 IMDb Location RDFS Schema
A.2.6 IMDb Photo RDFS Schema

![Diagram of IMDb Photo RDFS Schema]

imdb = <http://ifanzy.nl/Ontologies/IMDb/IMDbSchema#>

rdfs = <http://www.w3.org/2000/01/rdf-schema#>

A.2.7 IMDb Trailer RDFS Schema

![Diagram of IMDb Trailer RDFS Schema]

imdb = <http://ifanzy.nl/Ontologies/IMDb/IMDbSchema#>

rdfs = <http://www.w3.org/2000/01/rdf-schema#>

A.2.8 IMDb Episode RDFS Schema

![Diagram of IMDb Episode RDFS Schema]

imdb = <http://ifanzy.nl/Ontologies/IMDb/IMDbSchema#>

rdfs = <http://www.w3.org/2000/01/rdf-schema#>

time = <http://www.w3.org/2006/time#>
A.3 VideoDetective

Example of the metadata description of one trailer in the list returned by VideoDetective upon
the keyword query: ‘The Fifth Element’.

〈item〉
 〈Description〉
    New York cab driver, Corbin Dallas, didn’t mean to be a
    hero, but he just picked up the kind of fare that only
    comes around every five thousand years!
  </Description>
 〈Title〉THE FIFTH ELEMENT〈/Title〉
 〈Language〉 English〈/Language〉
 〈Country〉 United States of America〈/Country〉
 〈Studio〉Sony Pictures Home Entertainment〈/Studio〉
 〈StudioID〉21〈/StudioID〉
 〈Rating〉PG-13〈/Rating〉
 〈Genre〉Sci-Fi〈/Genre〉
 〈Warning〉Adult situations/language, violence〈/Warning〉
 〈Director〉Luc Besson〈/Director〉
 〈DirectorID〉5207〈/DirectorID〉
 〈Actor1〉Bruce Willis〈/Actor1〉
 〈ActorId1〉1005〈/ActorId1〉
 〈Actor2〉Gary Oldman〈/Actor2〉
 〈ActorId2〉2778〈/ActorId2〉
 〈Actor3〉Ian Holm〈/Actor3〉
 〈ActorId3〉3315〈/ActorId3〉
 〈Actor4〉Milla Jovovich〈/Actor4〉
 〈ActorId4〉5955〈/ActorId4〉
 〈Actor5〉Chris Tucker〈/Actor5〉
 〈ActorId5〉9289〈/ActorId5〉
 〈Link〉
  </Link〉
 〈Image〉
   http://videodetective.com/photos/166/007000_27.jpg〈/Image〉
  </videos>
 〈Link〉
   http://videodetective.com/photos/166/007000_27.jpg〈/Link〉
  </videos>
 〈Duration〉94〈/Duration〉
 〈IsHdSource〉False〈/IsHdSource〉
 〈videos〉
    〈video url="http://www.videodetective.net/player.aspx?
    cmd=6&fmt=5&videokbrate=80&publishedid=7001"
    medium="video" duration="94" Type="3gp" BitRate="80"
    StreamingFlavorId="23"/>
    ...
  </videos>
</item>
Bibliography


[137] A. Isaac and E. Summers. SKOS Simple Knowledge Organization System Primer. SKOS Primer Home Page (http://www.w3.org/TR/skos-primer/).


List of Figures

2.1 TV sales in the Netherlands (http://media.imnovator.nl/) ........................................... 12
2.2 Digital television penetration in Dutch households (http://media.imnovator.nl/) ............... 13
2.3 Digital television OpenTV application interface (Stoneroos Design Team) ....................... 14
2.4 Digital television MHP application interface (Stoneroos Design Team) .......................... 14
2.5 Traditional EPG grid view (http://www.team mediaportal.com/) .................................. 15
2.6 Different spaces for different needs .................................................................................... 19
2.7 Evolution in television capabilities ..................................................................................... 20
2.8 Example of MPEG-7 Metadata description ........................................................................ 24
2.9 TV-Anytime Metadata and content flow (Taken from ETSI TS 102 822-3-1) ..................... 27
2.10 Example of a MOS message ............................................................................................... 32
2.11 Example of Dublin Core properties in an RDF resource description ................................ 35
2.12 Metadata specifications overview ..................................................................................... 40
3.1 Example of an XML-TV Metadata description ................................................................ 49
3.2 Example of a ProgramGuideML Metadata description .................................................... 51
3.3 ProgramGuideML layer residing on top of TV-Anytime .................................................... 52
3.4 CRID resolving process ....................................................................................................... 53
3.5 The layout of the root os a TV-Anytime metadata document ............................................ 55
3.6 Example of a TV-Anytime Metadata description ................................................................ 56
3.7 Example of a part of the TV-Anytime content classification hierarchy ............................... 58
3.8 French course lessons overview ........................................................................................ 60
3.9 French course package structure ....................................................................................... 60
3.10 XML structure of a TV-Anytime described program .......................................................... 62
3.11 RDF Schema definition including an illustrative RDF instance ........................................ 64
4.1 Overview of our approach .................................................................................................... 68
5.1 Example of a TV-Anytime described program in RDF ....................................................... 80
5.2 GeoNames location instance in RDF ................................................................................ 82
5.3 TGN location instance in RDF .......................................................................................... 84
5.4 Part of the updated genre hierarchy ................................................................................... 87
5.5 Illustration of the TV-Anytime genre tree ......................................................................... 88
5.6 Knowledge integration in iFanzy ....................................................................................... 89
5.7 Linking Open Data cloud diagram as of July 2009 ............................................................ 91
5.8 Different Modeling approaches in XML-TV and TV-Anytime ............................................ 94
5.9 Example of an entry in the IMDb Actresses file ................................................................ 97
5.10 Example of an entry in the IMDb locations hierarchy ...................................................... 98
5.11 Movie instance enriched by IMDb .................................................................................... 99
5.12 Program metadata including a VideoDetective trailer ...................................................... 100
5.13 RDFS Schema of the main program modeling classes ...................................................... 102
5.14 Example instance exemplifying sources integration .......................................................... 104
5.15 iFanzy data sources and connections .............................................................................. 105
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>YouTube star rating dispersion (September 2009)</td>
<td>112</td>
</tr>
<tr>
<td>6.2</td>
<td>RDFS Structure of the User Model</td>
<td>114</td>
</tr>
<tr>
<td>6.3</td>
<td>Instance Example of the Property Class</td>
<td>115</td>
</tr>
<tr>
<td>6.4</td>
<td>Types of Assertions and possible Resources</td>
<td>116</td>
</tr>
<tr>
<td>6.5</td>
<td>Event class with associated context information</td>
<td>117</td>
</tr>
<tr>
<td>6.6</td>
<td>Event materialized in the user model</td>
<td>117</td>
</tr>
<tr>
<td>6.7</td>
<td>iFanzy assertions with target resources</td>
<td>118</td>
</tr>
<tr>
<td>6.8</td>
<td>User model assertions example</td>
<td>119</td>
</tr>
<tr>
<td>6.9</td>
<td>Newly deduced liking assertions</td>
<td>120</td>
</tr>
<tr>
<td>6.10</td>
<td>Example of the ASCII ARFF format</td>
<td>124</td>
</tr>
<tr>
<td>6.11</td>
<td>Naive Bayes network overview for user U</td>
<td>126</td>
</tr>
<tr>
<td>6.12</td>
<td>Naive Bayes probability calculation example</td>
<td>127</td>
</tr>
<tr>
<td>6.13</td>
<td>Social network distribution around the world in 2008 (The Ethority Weblog at <a href="http://www.ethority.net/blog/wp-content/uploads/2008/05/international_social_network_study.jpg">http://www.ethority.net/blog/wp-content/uploads/2008/05/international_social_network_study.jpg</a>)</td>
<td>132</td>
</tr>
<tr>
<td>6.15</td>
<td>A relevant part of John’s Hyves profile</td>
<td>133</td>
</tr>
<tr>
<td>7.1</td>
<td>The personalization process</td>
<td>142</td>
</tr>
<tr>
<td>7.2</td>
<td>Example of a SPARQL query finding inappropriate programs</td>
<td>147</td>
</tr>
<tr>
<td>8.1</td>
<td>Possible query decomposition (SeRQL)</td>
<td>161</td>
</tr>
<tr>
<td>9.1</td>
<td>The iFanzy server architecture</td>
<td>169</td>
</tr>
<tr>
<td>9.2</td>
<td>The client search interface</td>
<td>172</td>
</tr>
<tr>
<td>9.3</td>
<td>Example of a SPARQL query fetching television programs</td>
<td>174</td>
</tr>
<tr>
<td>9.4</td>
<td>The client interface showing program metadata</td>
<td>175</td>
</tr>
<tr>
<td>9.5</td>
<td>The iFanzy Home page (Stoneroos Design Team)</td>
<td>176</td>
</tr>
<tr>
<td>9.6</td>
<td>The iFanzy program details pane (Stoneroos Design Team)</td>
<td>177</td>
</tr>
<tr>
<td>9.7</td>
<td>The iFanzy set-top box login screen (Stoneroos Design Team)</td>
<td>178</td>
</tr>
<tr>
<td>9.8</td>
<td>The iFanzy set-top box PEPG screen (Stoneroos Design Team)</td>
<td>178</td>
</tr>
<tr>
<td>9.9</td>
<td>The iFanzy iPhone interface (Stoneroos Design Team)</td>
<td>180</td>
</tr>
<tr>
<td>10.1</td>
<td>Gender distribution in the evaluation (left) and general population (right)</td>
<td>192</td>
</tr>
<tr>
<td>10.2</td>
<td>Age distribution in the evaluation (left) and general population (right)</td>
<td>192</td>
</tr>
<tr>
<td>10.3</td>
<td>Education distribution in the evaluation (left) and general population (right)</td>
<td>192</td>
</tr>
<tr>
<td>10.4</td>
<td>How people initially perceived the recommendation scores</td>
<td>194</td>
</tr>
<tr>
<td>10.5</td>
<td>How people perceived the recommendation scores at the end of the test</td>
<td>194</td>
</tr>
<tr>
<td>10.6</td>
<td>User satisfaction evolution</td>
<td>194</td>
</tr>
<tr>
<td>10.7</td>
<td>Event type distribution</td>
<td>195</td>
</tr>
<tr>
<td>10.8</td>
<td>Rating value distribution</td>
<td>196</td>
</tr>
<tr>
<td>10.9</td>
<td>Average MAE evolution</td>
<td>197</td>
</tr>
<tr>
<td>10.10</td>
<td>Optimal influence for the five most relevant events</td>
<td>198</td>
</tr>
<tr>
<td>10.11</td>
<td>MAE for the 3 approaches in relation to the number of user ratings</td>
<td>204</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Audiovisual metadata specification overview ........................................ 38
2.2 Audiovisual metadata specification purposes ....................................... 39
3.1 Example of an Escort 2.4 program annotation ....................................... 49
5.1 Property overlap between XML-TV and TV-Anytime ............................... 95
5.2 Potentially relevant DBpedia resources in the television domain ............ 102
5.3 Triple sizes of the most important sources used .................................... 103
6.1 The most relevant features of a rating event ......................................... 123
6.2 Probability calculation result per class .............................................. 127
6.3 Entries in the historical program viewing statistics .............................. 129
6.4 Rules to transform historical viewing percentages to classes ................. 131
6.5 Entries in the historical program viewing statistics .............................. 131
8.1 Average execution times of typical iFanzy queries ................................. 160
8.2 Average execution times for two alternative query approaches of vertical decomposition with WordNet ......................................................... 162
8.3 IMDb size with respective query execution times ................................... 163
8.4 Comparison of execution times of an RDF repository with and without relational database optimizations ....................................................... 164
8.5 Execution results for free text search queries with and without index ......... 165
8.6 Average execution times for increasingly limited queries ....................... 165
8.7 Average execution times of typical iFanzy queries before and after optimizations ................................................................. 166
10.1 Numeral event statistics ................................................................. 195
10.2 Average user rating and number of new programs for our $B+ESA$ extension compared to $B+WO$ and $B$. Statistical significance and corresponding significance levels is indicated in brackets. ....................................................... 205
## Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>Standard Definition</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition</td>
</tr>
<tr>
<td>STB</td>
<td>Set-top Box</td>
</tr>
<tr>
<td>DVB</td>
<td>Digital Video Broadcasting</td>
</tr>
<tr>
<td>MHP</td>
<td>Multimedia Home Platform</td>
</tr>
<tr>
<td>iTV</td>
<td>interactive TV</td>
</tr>
<tr>
<td>EPG</td>
<td>Electronic Program Guide</td>
</tr>
<tr>
<td>PEPG</td>
<td>Personalized Electronic Program Guide</td>
</tr>
<tr>
<td>VOD</td>
<td>Video-On-Demand</td>
</tr>
<tr>
<td>PVR</td>
<td>Personal Video Recorder</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>CS</td>
<td>Classification Scheme</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>CRID</td>
<td>Content Reference IDentifier</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>DC</td>
<td>Dublin Core</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>CA</td>
<td>CRID Authority</td>
</tr>
<tr>
<td>MS</td>
<td>Metadata Service</td>
</tr>
<tr>
<td>XSD</td>
<td>XML Schema Document</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>RDFS</td>
<td>Resource Description Framework Schema</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>TGN</td>
<td>Getty Thesaurus of Geographic Names</td>
</tr>
<tr>
<td>SKOS</td>
<td>Simple Knowledge Organization System</td>
</tr>
<tr>
<td>IMDb</td>
<td>Internet Movie Database</td>
</tr>
<tr>
<td>FOAF</td>
<td>Friend Of A Friend</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>DF</td>
<td>Demographics Filtering</td>
</tr>
<tr>
<td>CBR</td>
<td>Case-Based Reasoning</td>
</tr>
<tr>
<td>CF</td>
<td>Collaborative Filtering</td>
</tr>
<tr>
<td>GWT</td>
<td>Google Web Toolkit</td>
</tr>
<tr>
<td>ESA</td>
<td>Explicit Semantic Analysis</td>
</tr>
</tbody>
</table>
Summary

In a variety of domains, developers nowadays are struggling with the dilemma of how they can provide a more personal service to their users. A more personal service can for example be facilitated by offering user-adapted search, generating recommendations, personalized content navigation, personalized user interfaces, etc. However, providing such functionality on top of a particular data set, requires a good knowledge of the relevant domain items (which can for example represent books, songs, TV programs, art pieces, etc.) as well as a good knowledge of the relevant users (in terms of the user’s behavior, interests, preferences, etc. with respect to those domain items).

In this dissertation and more specifically in Chapter 3 and Chapter 4, we describe the requirements and a domain-independent approach respectively, to provide context-sensitive and user-adapted access to heterogeneous data sources. This approach consists of three main parts, including: 1) Data Integration, 2) User Modeling and 3) User-Adapted Data Access.

Chapter 5 focusses on the integration of information from various heterogeneous data sources. To provide user-adapted access, a good description of the relevant domain items is key. The more descriptive information we have about every item, the more raw material is there to for example compare different items, compare items with user profiles, deduce new information, etc. Unfortunately, in the real world, items often come poorly described. On the other hand however, with the immense growth of available information on the Web, many different data sources (like IMDb, Wikipedia, social networks, etc.) exist and offer free access to their data. By using Semantic Web techniques we describe how we can enrich the descriptive metadata of those domain items by on the one hand integrating and matching instance metadata from different external sources, and on the other hand taking relevant ontological background information into account.

Chapter 6 concentrates on the second part in our approach: the creation of an extensive model of the end-user. Such a user model is the user’s digital representation and encompasses all valuable user data we can obtain. Information can be provided explicitly by the user himself (e.g. the user states that he is 45 years old, male, capable of speaking three languages, fond of tennis, etc.) but also implicitly. Implicit feedback includes all the information the user gives away without realizing it, by means of his behavioral patterns (e.g. the user watches the news every day at 8pm, he always adds books from the same author to his favorites, etc.). However, user feedback (both explicit and implicit) can be hard to interpret since it depends on a wide variety of parameters. Numerous influences like mood, location, time, environment, health, etc., make that people can behave very differently at any given time. Therefore, every statement in the user model is contextualized. In other words, the constrained setting in which a specific user statement was valid, which we call the statement’s context, is saved and is used later to accurately predict the user’s interests in any given situation. Further, since our approach depends on the quality and richness of the user model and new users usually start with an empty profile, we suffer from the so-called cold start problem. To deal with this situation, where new users have an empty profile, we provide a number of strategies based on user statistics and stereotypes to alleviate this problem.

The third and last part in our approach encompasses a strategy to adapt any user request and provide a personalized set of results, based on both the integrated data structure describing the domain and the user model. Chapter 7 describes a processing pipeline, consisting of three steps, which takes a user request as input and delivers a personal response. The first step of this strategy involves cleaning and conceptualizing the user’s request with respect to the current domain. Secondly, the updated query is sent to the database to retrieve matching results. However,
trying to find not only exactly matching results but also highly related results, the database automatically broadens the result space of the query in a controlled fashion. It does so by reasoning over well-chosen semantic relations including for example transitivity and synonymity. When matching results are retrieved, the last step filters them by following a set of rules. These rules are predefined and can include restrictions based on both ontological as well as user model information. This pipeline is employed both to provide user-adapted search as well as for the generation of personal recommendations.

However, systems dealing with potentially large amounts of data and on top of that provide complex functionality like reasoning, user-adapted search, integration of data, recommendations, etc., require extra care in terms of their database setup. Moreover, efficiency in terms of querying speed is vital for any such system’s long-term success. Therefore, in Chapter 8, we introduce a number of optimizations to improve the efficiency of the database in terms of size and querying speed.

To illustrate our approach, we apply it in the television domain which we introduce in Chapter 2. Together with Stoneroos we developed a cross-platform application called iFanzy, which tries to bring personalized access to television programs to the user via a set-top box interface, a Web site and an iPhone application. All three of these platforms are synchronized and behave as one ubiquitous application supporting the user in putting together the best possible television experience, by finding exactly those TV programs fitting the user best. In Chapter 9, we give an overview of these three platforms in terms of functionality and user interface. Furthermore, we perform an evaluation on the interface of the iFanzy Web portal including experiments like a Cognitive Walkthrough, the Thinking Aloud method and a Heuristic Evaluation.

Through the commercial availability of iFanzy we were able to further evaluate our approach, this time focusing on the recommendation quality and user satisfaction. In Chapter 10, we elucidate our evaluation which features a group of 60 people using the iFanzy Web interface for about two full weeks. From the data of this evaluation we investigate the influence of both explicit and implicit user feedback on the predictive power of the system and the accuracy of generated recommendations. To further improve the quality of the recommendation strategy, Chapter 10 concludes with an approach to improve the serendipity of the recommender system, which leads to more surprising or serendipitous discoveries. Reusing the data from the previous evaluation, we describe a number of measurements to quantify the degree of serendipity in the recommendations of a recommender system.
Samenvatting

In verschillende domeinen worstelen ontwikkelaars met het dilemma hoe ze aan hun gebruikers persoonlijker diensten kunnen aanbieden. Zo een persoonlijker dienst kan bijvoorbeeld aangeboden worden door middel van een adaptief zoekalgoritme, het genereren van aanbevelingen, persoonlijke contentnavigatie, persoonlijke gebruikersinterfaces, etc. Echter, het aanbieden van zulke functionaliteit bovenop een bepaalde dataset, vereist een goede kennis van zowel de relevante objecten in het domein (zoals boeken, liedjes, televisieprogramma’s, kunstobjecten, etc.) als de betrokken personen (in termen van persoonlijke gedrag, interesses, voorkeuren, etc. in relatie tot die objecten in het domein).

In deze dissertatie en meer specifiek in Hoofdstuk 3 en Hoofdstuk 4, beschrijven we respectievelijk de vereisten en een aanpak (onafhankelijke van het domein) voor het aanbieden van contextgevoelige en gepersonaliseerde toegang tot heterogene databronnen. Deze aanpak is opgebouwd uit drie centrale delen, waaronder: 1) Data integratie, 2) Modelleren van gebruikers en 3) Gepersonaliseerde datatoegang.

Hoofdstuk 5 richt zich op de integratie van informatie uit verschillende heterogene databronnen. Om de gebruiker gepersonaliseerde datatoegang aan te kunnen bieden, is een goede beschrijving van de relevante domeinobjecten essentieel. Hoe meer beschrijvende informatie beschikbaar voor elke object, hoe meer mogelijkheden we hebben om bijvoorbeeld verschillende objecten te vergelijken, objecten met gebruikersprofielen te vergelijken, nieuwe informatie af te leiden, enzovoort. Helaas, in de realiteit zijn zulke domeinobjecten vaak slecht beschreven. Aan de andere kant, gezien de enorme groei van de beschikbare informatie op het Web, zijn er veel verschillende gegevensbronnen (zoals IMDb, Wikipedia, sociale netwerken, etc.) ontstaan die gratis toegang tot hun gegevens aanbieden. Door gebruik te maken van technieken van het Semantische Web, beschrijven we hoe we de metadata van die domeinobjecten kunnen verrijken door aan de ene kant informatie uit verschillende externe bronnen te integreren, en aan de andere kant relevante ontologische achtergrondinformatie mee te nemen.

Hoofdstuk 6 concentreert zich op het tweede deel van onze aanpak: het creëren van een uitgebreid model van de gebruiker. Zo een gebruikersmodel is een digitale representatie en omvat alle waardevolle gegevens van de gebruiker die we kunnen verkrijgen. Informatie kan expliciet door de gebruiker verstrekt worden (bijvoorbeeld wanneer de gebruiker verklaart dat hij een 45 jaar oude man is, drie talen spreekt, van tennis houdt, enz.), maar ook impliciet. Impliciete feedback omvat alle informatie die de gebruiker weggeeft zonder het te beseffen, bijvoorbeeld door middel van zijn gedragspatronen (de gebruiker kijkt iedere dag om 8 uur naar het nieuws, boeken van een bepaalde auteur voegt hij altijd toe aan zijn favorieten, enz.). Echter, gebruikersfeedback (zowel expliciet als impliciet) is soms moeilijk te interpreteren omdat het afhankelijk is van een grote verscheidenheid aan externe factoren. Tal van invloeden zoals de huidige stemming, locatie, tijd, omgeving, gezondheid, enzovoort, maken dat mensen zich heel verschillend kunnen gedragen op elk gegeven moment. Daarom wordt elke toevoeging in het gebruikersmodel gecontextualisert. Met andere woorden, de beperkte situatie, die we “de context” noemen, waarin een specifieke uitdrukking geldig was, wordt behandeld zodat die later gebruikt kan worden om nauwkeuriger voorspellen welke de gebruiker zou kunnen interesseren in een bepaalde gegeven situatie. Verder, aangezien onze aanpak afhankelijk is van de kwaliteit en de rijkdom van het gebruikersmodel en nieuwe gebruikers meestal starten met een leeg profiel, lijden we onder het zogenaamde koude start probleem. Om deze situatie, waarbij nieuwe gebruikers een leeg profiel hebben, aan te pakken
stellen we een aantal strategieën voor, op basis van gebruikersstatistieken en stereotypen, die het probleem verlichten.

Het derde en laatste deel van onze aanpak omvat een strategie nodig om elke vraag van de gebruiker aan te passen zodat we een persoonlijke set van resultaten, gebaseerd op zowel de geïntegreerde datastructuur die het domein beschrijft als het gebruikersmodel, terug kunnen geven. Hoofdstuk 7 beschrijft een proces, bestaande uit drie stappen, dat een vraag van de gebruiker als input neemt en resulteert in een gepersonaliseerd antwoord. De eerste stap omvat het zuiveren en conceptualiseren van de gebruikersvraag met betrekking tot het huidige domein. Als tweede stap wordt de bijgewerkte query naar de database verstuurdd om resultaten, met betrekking tot de gestelde vraag, op te halen. Echter, in een poging om niet alleen exacte antwoorden maar ook zeer verwante resultaten te verkrijgen, verbreedt de database automatisch de resultaatsruimte van de query op een gecontroleerde manier. Het doet dit door te redeneren over goed gekozen semantische relaties, zoals de transitiviteits- en synoniemrelatie. Wanneer de corresponderende resultaten worden opgehaald, worden deze in de laatste stap gefilterd door een set van regels. Deze regels zijn vooraf gedefinieerd, en kunnen beperkingen op basis van zowel ontologische structuren als informatie uit het gebruikersmodel bevatten. Dit proces is zowel werkzaam op gepersonaliseerde zoekacties alsook voor de generatie van persoonlijke aanbevelingen.

Echter, systemen die potentieel grote hoeveelheden informatie bevatten en daarbovenop complexe functionaliteiten zoals redeneren, gepersonaliseerd zoeken, integratie van data, aanbevelingen, enz., aanbieden, vereisen extra inspanningen met betrekking tot het opzetten van de nodige databasestructuren. Meer zelfs, de efficiëntie en snelheid van het uitvoeren van queries op de database is op lange termijn van vitaal belang voor het succes van elk zulk systeem. In Hoofdstuk 8 introduceren we daarom een aantal optimalisatietechnieken om de efficiëntie van de database te verbeteren met betrekking tot de grootte en snelheid van de verschillende datasets.

Ter illustratie hebben we onze aanpak toegepast in het televisiedomein, welk we introduceren in Hoofdstuk 2. Samen met Stoneroos ontwikkelden we een crossplatform applicatie genaamd iFanzy, die prototype gepersonaliseerde toegang tot televisieprogramma's aan te bieden aan de gebruiker via een set-top box interface, een website en een iPhone applicatie. Deze drie platformen zijn allen gesynchroniseerd en gedragen zich als één alomtegenwoordige applicatie die de gebruiker ondersteunt bij het samenstellen van de beste televisie-ervaring, door het zoeken van juist die televisie programma's die het beste passen bij de gebruiker. In Hoofdstuk 9 geven we een overzicht van deze drie platformen in termen van functionaliteit en gebruikersinterface. Vervolgens evalueren we de interface van de iFanzy website door het uitvoeren van experimenten zoals een Cognitive Walkthrough, de Thinking Aloud method en een Heuristic Evaluation.

Dankzij de commerciële beschikbaarheid van iFanzy waren we in staat onze aanpak verder te evalueren, echter nu specifiek gericht op de kwaliteit van de aanbevelingen en de tevredenheid van de gebruiker. In Hoofdstuk 10, lichten we deze evaluatie, waarin een groep van 60 mensen de iFanzy Web interface voor ongeveer twee volle weken gebruikten, verder toe. Uit de gegevens van deze evaluatie onderzoeken we de invloed van expliciete en impliciete gebruikersfeedback op de voorspellende kracht van het systeem en de nauwkeurigheid van de gegenereerde aanbevelingen. Om de kwaliteit van de aanbevelingsstrategie verder te verbeteren, sluiten we Hoofdstuk 10 af met een techniek voor het verhogen van de serendipiteit van het aanbevelingssysteem, dat tot meer verrassende of serendipiteuze ontdekkingen leidt. Door het hergebruiken van de gegevens van de eerder besproken evaluatie, beschrijven we een aantal metingen om de mate van serendipiteit in de aanbevelingen van ons systeem te kwantificeren.
Pieter Bellekens was born in Mechelen, Belgium, on November 3, 1980. In 1998 he graduated high school with specialization in applied sciences. In the same year, he started studying at the De Nayer Institute, where he received his first master degree with specialization in electronics and computer science in 2002. Later that year, he moved to the Netherlands, to continue studying in Eindhoven at the University of Technology. Two years later he obtained his second master degree, with specialization mathematics and computer science, and graduated cum laude in the group of Geert-Jan Houben. There, they worked within the Hera project which focussed on personalization and data integration in Web Information Systems.

Between 2005 and 2010 Pieter Bellekens was a PhD candidate at Eindhoven University of Technology, in the department of mathematics and computer science. Within the first two years, research was performed within the ITEA Passepartout project (aimed at coupling home media-centers to home networks for rendering scalable and personalized TV content on different devices in a seamless fashion) supervised by Lora Aroyo and in close collaboration with Stoneroos Interactive Television. Since 2007 he was also employed at Stoneroos as head of research.

His research interests include novel and innovative multimedia applications, information integration, user modeling, personalization, Web information systems, Semantic Web technologies and ubiquitous and context-aware computing.
SIKS Dissertatiereeks

1998

1998-1 Johan van den Akker (CWI)
DEGAS - An Active Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W. Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling: Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

2000

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UvA)
Sociaal-organisatorische gevolgen van kennis-technologie: een procesbenadering en actorperspectief.

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coup (EUR)
Sensitivity Analysis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

2001

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UVa)
Learning as problem solving

2001-4 Evgeni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization
2001-8 Pascal van Eck (VU)  
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 Pieter Jan ’t Hoen (RUL)  
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)  
Modeling and Simulating Work Practice  
BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VUA)  
Knowledge Management: The Role of Mental Models in Business Systems Design

2002-01 Nico Lassing (VU)  
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwal (UT)  
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)  
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castello Valdueza (UU)  
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)  
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)  
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)  
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijn (VU)  
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willems-Jan van den Heuvel(KUB)  
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)  
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)  
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (Uva)  
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)  
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)  
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2003-01 Heiner Stuckenschmidt (VU)  
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)  
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)  
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)  
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA)  
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)  
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)  
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)  
Repair Based Scheduling

2003-09 Rens Kortmann (UM)  
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)  
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)  
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)  
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)  
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)  
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital MediaWarehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)
Learning Search Decisions

2004

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs, voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek (UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politieke gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)
Using generative probabilistic models for multimedia retrieval

2004-19 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

2005

2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM)
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)
Adaptive Game AI

2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-09 Joem Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UVA)
Explanation Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry
2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
2005-14 Boris Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes
2005-16 Joris Graaumans (UU)
Usability of XML Query Languages
2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components
2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks
2005-19 Michel van Dartel (UM)
Situated Representation
2005-20 Cristina Coteanu (UL)
Cyber Consumer Law, State of the Art and Perspectives
2005-21 Wijnand Derks (UT)
Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006

2006-01 Samuil Angelov (TUE)
Foundations of B2B Electronic Contracting
2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations
2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems
2006-04 Marta Sabou (VU)
Building Web Service Ontologies
2006-05 Cees Pierik (UU)
Validation Techniques for Object-Oriented Proof Outlines
2006-06 Ziv Baida (VU)
Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling
2006-07 Marko Smiljanic (UT)
XML schema matching – balancing efficiency and effectiveness by means of clustering
2006-08 Eelco Herder (UT)
Forward, Back and Home Again - Analyzing User Behavior on the Web
2006-09 Mohamed Wahdan (UM)
Automatic Formulation of the Auditor’s Opinion
2006-10 Ronny Siebes (VU)
Semantic Routing in Peer-to-Peer Systems
2006-11 Joeri van Ruth (UT)
Flattening Queries over Nested Data Types
2006-12 Bert Bongers (VU)
Interactivation - Towards an e-ology of people, our technological environment, and the arts
2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents
2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain
2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks
2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device
2006-18 Valentin Zhizhukh (UVA)
Graph transformation for Natural Language Processing
2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach
2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining
2006-21 Bas van Gils (RUN)
Aptness on the Web
2006-22 Paul de Vrieze (RUN)
Fundamentals of Adaptive Personalisation
2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web
2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources
2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC
2006-26 Vojkan Mihailovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval
2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories
2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval
2007

2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures

2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach

2007-03 Peter Mila (VU)
Social Networks and the Semantic Web

2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach

2007-05 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance

2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs

2007-07 Natasa Jovanovic' (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings

2007-08 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations

2007-09 David Mobach (VU)
Agent-Based Mediated Service Negotiation

2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols

2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System

2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty

2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology

2007-14 Nick Bergboer (UM)
Context-Based Image Analysis

2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model

2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems

2007-17 Theodore Charitos (UU)
Reasoning with Dynamic Networks in Practice

2007-18 Bart Orriens (UvT)
On the development an management of adaptive business collaborations

2007-19 David Levy (UM)
Intimate relationships with artificial partners

2007-20 Slinger Jansen (UU)
Customer Configuration Updating in a Software Supply Network

2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005

2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE)
Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramirez Camps (CWI)
Structural Features in XML Retrieval

2007-25 Joost Schalcken (VU)
Empirical Investigations in Software Process Improvement

2008

2008-01 Katalin Boer-Sorbn (EUR)
Agent-Based Simulation of Financial Markets: A modular, continuous-time approach

2008-02 Alexei Sharpanskykh (VU)
On Computer-Aided Methods for Modeling and Analysis of Organizations

2008-03 Vera Hollik (VUA)
Optimizing hierarchical menus: a usage-based approach

2008-04 Ander de Keijzer (UT)
Management of Uncertain Data - towards unattended integration

2008-05 Bela Mutschler (UT)
Modeling and simulating causal dependencies on process-aware information systems from a cost perspective

2008-06 Arjen Hommersom (RUN)
On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective

2008-07 Peter van Rosmalen (OU)
Supporting the tutor in the design and support of adaptive e-learning

2008-08 Janneke Bolt (UU)
Bayesian Networks: Aspects of Approximate Inference

2008-09 Christof van Nimwegen (UU)
The paradox of the guided user: assistance can be counter-effective

2008-10 Wauter Bosma (UT)
Discourse oriented summarization

2008-11 Vera Kartseva (VU)
Designing Controls for Network Organizations: A Value-Based Approach
2008-12 Jozsef Farkas (RUN)
A Semiotically Oriented Cognitive Model of Knowledge Representation
2008-13 Caterina Carraciolo (UVA)
Topic Driven Access to Scientific Handbooks
2008-14 Arthur van Bunningen (UT)
Context-Aware Querying; Better Answers with Less Effort
2008-15 Martijn van Otterlo (UT)
2008-16 Henriette van Vugt (VU)
Embodied agents from a user’s perspective
2008-17 Martin Op’t Land (TUD)
Applying Architecture and Ontology to the Splitting and Allying of Enterprises
2008-18 Guido de Croon (UM)
Adaptive Active Vision
2008-19 Henning Rode (UT)
From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search
2008-20 Rex Arendsen (UVA)
Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven
2008-21 Krisztian Balog (UVA)
People Search in the Enterprise
2008-22 Henk Koning (UU)
Communication of IT-Architecture
2008-23 Stefan Visscher (UU)
Bayesian network models for the management of ventilator-associated pneumonia
2008-24 Zharko Aleksovski (VU)
Using background knowledge in ontology matching
2008-25 Geert Jonker (UU)
Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency
2008-26 Marijn Huijbregts (UT)
Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled
2008-27 Hubert Vogten (OU)
Design and Implementation Strategies for IMS Learning Design
2008-28 Ildiko Flesch (RUN)
On the Use of Independence Relations in Bayesian Networks
2008-29 Dennis Reidsma (UT)
Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans
2008-30 Wouter van Atteveldt (VU)
Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content
2008-31 Loes Braun (UM)
Pro-Active Medical Information Retrieval
2008-32 Trung H. Bai (UT)
Toward Affective Dialogue Management using Partially Observable Markov Decision Processes
2008-33 Frank Terpstra (UVA)
Scientific Workflow Design; theoretical and practical issues
2008-34 Jeroen de Knijf (UU)
Studies in Frequent Tree Mining
2008-35 Ben Torben Nielsen (UvT)
Dendritic morphologies: function shapes structure

2009

2009-01 Rasa Jurgelenaite (RUN)
Symmetric Causal Independence Models
2009-02 Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques
2009-03 Hans Stoel (UvT)
A Framework for Evidence-based Policy Making Using IT
2009-04 Josephine Nabukunya (RUN)
Improving the Quality of Organisational Policy Making using Collaboration Engineering
2009-05 Sietse Overbeek (RUN)
Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality
2009-06 Muhammad Subianto (UU)
Understanding Classification
2009-07 Ronald Poppe (UT)
Discriminative Vision-Based Recovery and Recognition of Human Motion
2009-08 Volker Nannen (VU)
Evolutionary Agent-Based Policy Analysis in Dynamic Environments
2009-09 Benjamin Kanaga (RUN)
Design, Discovery and Construction of Service-oriented Systems
2009-10 Jan Wieliemaker (UVA)
Logic programming for knowledge-intensive interactive applications
2009-11 Alexander Boer (UVA)
Legal Theory, Sources of Law & the Semantic Web
2009-12 Peter Masseth (TUE, Humboldt-Universitaet zu Berlin)
Operating Guidelines for Services
2009-13 Steven de Jong (UM)
Fairness in Multi-Agent Systems
2009-14 Maksym Korotkiy (VU)
From ontology-enabled services to service-enabled ontologies
(making ontologies work in e-science with ONTO-SOA)
2009-15 Rinke Hoekstra (UVA)
Ontology Representation - Design Patterns and Ontologies that Make Sense
2009-16 Fritz Reul (UvT)
New Architectures in Computer Chess
2009-17 Laurens van der Maaten (UvT)
Feature Extraction from Visual Data
2009-18 Fabian Groffen (CWI)
Armada, An Evolving Database System
2009-19 Valentijn Robu (CWI)
Modeling Preferences. Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets
2009-20 Bob van der Vecht (UU)
Adjustable Autonomy: Controlling Influences on Decision Making
2009-21 Stijn Vanderlooy (UM)
Ranking and Reliable Classification
2009-22 Pavel Serdyukov (UT)
Search For Expertise: Going beyond direct evidence
2009-23 Peter Hofgesang (VU)
Modelling Web Usage in a Changing Environment
2009-24 Annerieke Heuwinkel (VUA)
Cognitive Models for Training Simulations
2009-25 Alex van Ballegooij (CWI)
"RAM: Array Database Management through Relational Mapping"
2009-26 Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services
2009-27 Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web
2009-28 Sander Evers (UT)
Sensor Data Management with Probabilistic Models
2009-29 Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications
2009-30 Marcin Zukowski (CWI)
Balancing vectorized query execution with bandwidth-optimized storage
2009-31 Sofya Katrenko (UVA)
A Closer Look at Learning Relations from Text
2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)
Architectural Knowledge Management: Supporting Architects and Auditors
2009-33 Khiet Truong (UT)
How Does Real Affect Affect Affect Recognition In Speech?
2009-34 Inge van de Weerd (UU)
Advancing in Software Product Management: An Incremental Method Engineering Approach
2009-35 Wouter Koelwijnt (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling
2009-36 Marco Kalz (OUN)
Placement Support for Learners in Learning Networks
2009-37 Hendrik Dracholer (OUN)
Navigation Support for Learners in Informal Learning Networks
2009-38 Riina Vuorikari (OU)
Tags and self-organisation: a metadata ecology for learning resources in a multilingual context
2009-39 Christian Stahl (TUE, Humboldt-Universitat zu Berlin)
Service Substitution – A Behavioral Approach Based on Petri Nets
2009-40 Stephan Raaijmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language
2009-41 Igor Berezhnyy (UvT)
Digital Analysis of Paintings
2009-42 Toine Bogers
Recommender Systems for Social Bookmarking
2009-43 Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients
2009-44 Roberto Santana Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations
2009-45 Jilles Vreeken (UU)
Making Pattern Mining Useful
2009-46 Loredana Afanasiev (UvA)
Querying XML: Benchmarks and Recursion

====
2010
====
2010-01 Matthijs van Leeuwen (UU)
Patterns that Matter
2010-02 Ingo Wassink (UT)
Work flows in Life Science
2010-03 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents
2010-04 Olga Kulyk (UT)
Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments
<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-05</td>
<td>Claudia Hauff (UT)</td>
<td>Predicting the Effectiveness of Queries and Retrieval Systems</td>
</tr>
<tr>
<td>2010-06</td>
<td>Sander Bakkes (UvT)</td>
<td>Rapid Adaptation of Video Game AI</td>
</tr>
<tr>
<td>2010-07</td>
<td>Wim Fikkert (UT)</td>
<td>Gesture interaction at a Distance</td>
</tr>
<tr>
<td>2010-08</td>
<td>Krzysztof Siewicz (UL)</td>
<td>Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments</td>
</tr>
<tr>
<td>2010-09</td>
<td>Hugo Kielman (UL)</td>
<td>A Politieke gegevensverwerking en Privacy. Naar een effectieve waarborging</td>
</tr>
<tr>
<td>2010-10</td>
<td>Rebecca Ong (UL)</td>
<td>Mobile Communication and Protection of Children</td>
</tr>
<tr>
<td>2010-11</td>
<td>Adriaan Ter Mors (TUD)</td>
<td>The world according to MARP: Multi-Agent Route Planning</td>
</tr>
<tr>
<td>2010-12</td>
<td>Susan van den Braak (UU)</td>
<td>Sensemaking software for crime analysis</td>
</tr>
<tr>
<td>2010-13</td>
<td>Gianluigi Folino (RUN)</td>
<td>High Performance Data Mining using Bio-inspired techniques</td>
</tr>
<tr>
<td>2010-14</td>
<td>Sander van Splunter (VU)</td>
<td>Automated Web Service Reconfiguration</td>
</tr>
<tr>
<td>2010-15</td>
<td>Lianne Bodenstaff (UT)</td>
<td>Managing Dependency Relations in Inter-Organizational Models</td>
</tr>
<tr>
<td>2010-16</td>
<td>Sicco Verwer (TUD)</td>
<td>Efficient Identification of Timed Automata, theory and practice</td>
</tr>
<tr>
<td>2010-17</td>
<td>Spyros Kotsoulas (VU)</td>
<td>Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications</td>
</tr>
<tr>
<td>2010-18</td>
<td>Charlotte Gerritsen (VU)</td>
<td>Caught in the Act: Investigating Crime by Agent-Based Simulation</td>
</tr>
<tr>
<td>2010-19</td>
<td>Henriette Craner (UvA)</td>
<td>People’s Responses to Autonomous and Adaptive Systems</td>
</tr>
<tr>
<td>2010-20</td>
<td>Ivo Swartjes (UT)</td>
<td>Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative</td>
</tr>
<tr>
<td>2010-21</td>
<td>Harold van Heerde (UT)</td>
<td>Privacy-aware data management by means of data degradation</td>
</tr>
<tr>
<td>2010-22</td>
<td>Michiel Hildebrand (CWI)</td>
<td>End-user Support for Access to Heterogeneous Linked Data</td>
</tr>
<tr>
<td>2010-23</td>
<td>Bas Steunebrink (UU)</td>
<td>The Logical Structure of Emotions</td>
</tr>
<tr>
<td>2010-24</td>
<td>Dmytro Tykhonov (CWI)</td>
<td>Designing Generic and Efficient Negotiation Strategies</td>
</tr>
<tr>
<td>2010-25</td>
<td>Zulfiqar Ali Memon (VU)</td>
<td>Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective</td>
</tr>
<tr>
<td>2010-26</td>
<td>Ying Zhang (CWI)</td>
<td>XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines</td>
</tr>
<tr>
<td>2010-27</td>
<td>Marten Voulon (UL)</td>
<td>Automatisch contracteren</td>
</tr>
<tr>
<td>2010-28</td>
<td>Arne Koopman (UU)</td>
<td>Characteristic Relational Patterns</td>
</tr>
<tr>
<td>2010-29</td>
<td>Stratos Idreos(CWI)</td>
<td>Database Cracking: Towards Auto-tuning Database Kernels</td>
</tr>
<tr>
<td>2010-30</td>
<td>Marieke van Erp (UvT)</td>
<td>Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval</td>
</tr>
<tr>
<td>2010-31</td>
<td>Victor de Boer (UvA)</td>
<td>Ontology Enrichment from Heterogeneous Sources on the Web</td>
</tr>
<tr>
<td>2010-32</td>
<td>Marcel Hiel (UvT)</td>
<td>An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems</td>
</tr>
<tr>
<td>2010-33</td>
<td>Robin Aly (UT)</td>
<td>Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval</td>
</tr>
<tr>
<td>2010-34</td>
<td>Teduh Birghahyu (UT)</td>
<td>Interaction Design in Service Compositions</td>
</tr>
<tr>
<td>2010-35</td>
<td>Dolf Trieschnigg (UT)</td>
<td>Proof of Concept: Concept-based Biomedical Information Retrieval</td>
</tr>
<tr>
<td>2010-36</td>
<td>Jose Janssen (OU)</td>
<td>Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification</td>
</tr>
<tr>
<td>2010-37</td>
<td>Niels Lohmann (TUE)</td>
<td>Correctness of services and their composition</td>
</tr>
<tr>
<td>2010-38</td>
<td>Dirk Fahland (TUE)</td>
<td>From Scenarios to components</td>
</tr>
<tr>
<td>2010-39</td>
<td>Ghazanfar Farooq Siddiqui (VU)</td>
<td>Integrative modeling of emotions in virtual agents</td>
</tr>
<tr>
<td>2010-40</td>
<td>Mark van Assem (VU)</td>
<td>Converting and Integrating Vocabularies for the Semantic Web</td>
</tr>
<tr>
<td>2010-41</td>
<td>Guillaume Chaillot (UM)</td>
<td>Monte-Carlo Tree Search</td>
</tr>
<tr>
<td>2010-42</td>
<td>Sybren de Kinderen (VU)</td>
<td>Needs-driven service bundling in a multi-supplier setting - the computational c3-service approach</td>
</tr>
<tr>
<td>2010-43</td>
<td>Peter van Kranenburg (UU)</td>
<td>A Computational Approach to Content-Based Retrieval of Folk Song Melodies</td>
</tr>
</tbody>
</table>