Music Recommendation: a multi-faceted approach

by

Òscar Celma

Submitted in partial fulfilment of the requirements for
the degree of Diploma of Advanced Studies
Doctorate in Computer Science and Digital Communication

Advisor: Xavier Serra i Casals

Department of Technology
Universitat Pompeu Fabra

Barcelona, July 2006

"Hey mister record man,
the joke's on you.
Running your label
like it was 1992.
Hey mister record man,
your system can't compete,
It's the new artist model,
file transfer complete."
— Download this Song. MC Lars (2006)

Abstract

This research project is about music recommendation. More precisely, it is concerned with the problems that appear when computer programs try to automatically recommend music assets to users. The aim of this work is to present a multi–faceted approach to the music recommendation problem.

Music recommendation involves the modelling of user preferences, as well as the matching between profiles and music related information. In recent years the typical music consumption behaviour has changed dramatically. Personal music collections have grown favoured by technological improvements in networks, storage, portability of devices and Internet services. This thesis presents the current methods used to recommend music assets, and reviews also some proposals of modelling user's musical preferences.

Furthermore, the characterization of the music objects to be recommended is a complex task. This thesis studies the different facets of music knowledge management (based on editorial, cultural and acoustic metadata). This holistic approach allows to describe the different components of the music objects. These descriptions permit to enhance and improve music recommendations. Finally, the descriptions are enclosed into an ontological framework for semantic integration and retrieval of audiovisual metadata.

As a test–bed example, two prototypes have been developed: a music search engine and music discovery based on music similarity, and a hybrid music recommender. In both cases, the systems exploit and crawl music related content from the Web.

Acknowledgments

First of all, I would like to thank Dr. Xavier Serra, my supervisor, for giving me the opportunity to work on this very fascinating topic, at the Music Technology Group. Also, and specially, I want to thank Perfecto Herrera for providing countless suggestions and support for my work in this research project.

I would like to thank all the people whom I have been involved with, in all the (mostly EU) research projects. A non–exhaustive list includes: David Garcia, Emilia Gómez, Sebastian Streich, Bee Suan, Pedro Cano and Koppi. And, I also want to thank the other people indirectly involved in the SIMAC research project for their appreciated cooperation. Further thanks go to my colleagues from Office 320: Pau Arumí and Maarten de Boer.

I am very grateful to a loooong list of MTG members. Some of them are: the Singing Voice Group —Alex Loscos, Jordi Bonada, Oscar Mayor, Jordi Janer and Lars Fabig— where I started when I joined MTG back in 2000, Gunnar Holmberg —for proof reading—, Jose Pedro, Bram de Jong, Esteban Maestre —for sharing the DEA crazyness—, as well as the Administration Staff (Cristina, Joana and Salvador), and the *sysadmins* (Maarten de Boer, Ramón Loureiro, and Carlos Atance), who provided help, hints and patience when I played around with *their* machines.

Also, I would like to thank the old–colleagues that helped me in some bits of the thesis: Roberto Garcia, Xavier Oliver, Nicolas Falquet, Miquel Ramirez, Enric Mieza, and Timothy John Taylor —for that music inspiration.

This research was funded by a scholarship from Universitat Pompeu Fabra and by the EU projects: EU-FP6-IST-507142 SIMAC and Opendrama IST-2000-28197.

Last but not least, this work would have never been possible without the encouragement of my wife Claudia, who has provided me love and patience, my lovely son Alex, my parents Tere and Toni, my brother Marc and my sister in law Marta, and all the family from Mexico. Finally, they will understand what my work is about... hopefully.

To Claudia and Alex.
"Two hearts beat as one".

Contents

Chapter 1

Introduction

This chapter contains some remarks on the background, context, and motivation of the research presented. Also, the outline of the thesis is presented, while a concise demarcation of the topic under research will be presented in the following chapters.

1.1 About the author

I obtained the Spanish master degree in Computer Science from Universitat Politècnica de Catalunya, in 2001. It was one year before, in February 2000, when I joined in the Music Technology Group (MTG), a research group of the Universitat Pompeu Fabra (UPF), located in Barcelona. I did the Master Thesis during that year. My work was centered on creating a musical expression model for a singing voice synthesizer.

After graduation, I was employed by the Music Techonology Group as a research engineer, and I am enrolled in the PhD program in Computer Science and Digital Communication at UPF, and holds a University scholarship.

Since 2000 I was been involved in several projects at MTG. I started working with the *Singing Voice Group* in conjunction with the *Yamaha* company. My work was focused on musical expression models applied to a singing voice synthesizer. After that, from 2002 to 2004 I was involved in the Opendrama IST-2000-28197 Project. The project aim was the development and integration of a novel platform to author and to deliver rich cross-media digital objects, in the opera domain. In the project, I worked on the multimedia standards field

to describe audiovisual assets. The objective was to describe and manage music descriptions using the MPEG-7 standard. One of the main outputs of the project was a multimedia database that contained opera descriptions based on the MPEG-7 standard. Since 2004, I worked for the EU-FP6-IST-507142 project SIMAC (Semantic Interaction with Music Audio Contents). The project's main goal was the automatic generation of semantic descriptors and the development of prototypes for exploration, recommendation, and retrieval of music[1]. The project ended in April, 2006.

I am involved in teaching in the Department of Technology of the UPF, since 2001. I have been teaching several Databases' courses as well as basic courses related with Data Structures.

I am also an amateur musician with classic guitar education. Yet, I also enjoy to make some noise with my Gretsch electric guitar, as well as playing wih some friends, from time to time.

1.2 A brief story

Back in the 80's, my friend *M* was a passionate for the music, a true music lover. M used to buy tens of vynils each month. He had his favorite record store where he bought most of the music. After a few years, he became a friend of the sellerman, called *J*. They knew very well each other. They got an interesting music–relationship. Every time J received some material straight from UK, he set aside a few vynils he thought M would like. M also listened to some great music through one —not very mainstream— radio station, and discovered artists via a paper magazine, *P1*. He used to go to a lot of gigs, that were announced on the radio or advertised on the P1 magazine. After the concert, a couple of T–shirts from the merchandise were added to his collection. Moreover, a tape with the recorded gig was added, too, to his list of his hand–made bootlegs.

One day he told me how he discovered his all time most favorite band, *TDA*. M saw the cover artwork of an album. It's great! he said. M asked J about the band, but he did not know anything, nor had heard of TDA before. Anyway, M liked the cover, and he bought the album. He spent the whole weekend listening to the album. He felt in love with the band. After a few listenings, he linked them

[1]Further information on the project can be found at URL http://www.semanticaudio.org

with other bands he liked: *RS*, *F* and *NYD*. In the late 80's, when TDA released the next album, they started to be a bit famous in UK. They even appeared on the (now defunct) *TOTP*, in '89. But it was very difficult for M to track what TDA were doing (e.g recording, touring, rehearsing). It was very difficult to get this information.

In 1996, M met his wife in a Rock Festival. They felt in love while chating about music, bands, and expressing what their favorite songs meant to each one. Strangely enough, they shared some songs altogether. Although the songs reminded them different situations and events, they got the feeling that a few songs meant somehow the same to them. In 2002, during the wedding, J prepared a special CDr for them. The people who know them, said that the playlist was a perfect resumé of M's life. Two years later, M found the perfect lullaby tune for his baby The melody was a slow version of one of his favorites song from TDA. That tune *automatically* made his son felt sleep.

Back to the 90's, when things changed a bit, M started to buy CDs and forgot about the old vynils. He never discovered any other good band based on the artwork (he never really liked that small CD covers!). Regarding TDA, they incrementaly got lost in the middle of the mainstream pop–punk, Nirvana's, and the like. Since then, their record company, *CR*, treated them as a Rock Stars: TDA got everything they wanted to. But all of a sudden, in 1993, CR thought the band did not evolved in all those years, and they were fired. A few years later, *T*, the main singer and composer started from scratch. He created a couple of nice albums with a small record company. But he did not received any money from them.

A few years later, in 1998, M re–discovered the band and T searching on Internet. There were a few webpages, and a mailing list —with around 200 members— talking about them. M knew then that T recorded some albums and he was able to purchase them. After a few bad experiences with the record industry, T decided to do everything on his own: recording an album, mixing, doing the layout and artwork, distributing it through Internet, and a long etcetera. Fans could get the CD directly from the artist or buying it from a couple of on-line independent record stores. Via the mailing list, M discovered a lot of related artists, too. He could get their music by trading CDr's with the members of the list, or purchasing the albums via on–line music sellers.

Nowadays, T is doing fine, but he is a stressed artist: he has to manage

his myspace.com page, deal with his paypal.com account, organise the whole European Tour with his new band, manage the bookings, pay the musicians, etc. Now, he cannot find the time to compose a 3–chords like song! Still, he likes to be able to control the whole artistic process of his work. And he gets more money than dealing with record companies. On the other hand, M started to understand how to exploit the net: he downloads lots of music via p2p applications, MP3-weblogs, podcast sessions, etc. He has more than one thousand CDr's with digital audio files. But M has not enough time to listen to his CDr's nor his external hard disks with about 120Gb of audio files. M says he has not listened to most of the music he has downloaded. M does not listen to the *traditional* radio anymore, neither buys P1 magazine. M says he does not discover artists as he used to. Yet, M does not speak with J anymore. M and J had a big fight when J knew that M won the eBay auction both were bidding to. It was one of the most wanted items J was looking for to finish his *RS* collection. Last week, M got divorced and lost all his music collection. His ex sold it to J. Now, all his music belongings are a digital pile of CDr's, and a couple of external hard disks.

And now, let's move to serious bit of the story...

1.3 Background

In recent years the typical music consumption behaviour has changed dramatically. Personal music collections have grown favoured by technological improvements in networks, storage, portability of devices and Internet services. The amount and availability of songs has de-emphasized its value: it is usually the case that users own many digital music files that they have only listened to once or even never. It seems reasonable to think that by providing listeners with efficient ways to create a personalized order on their collections, and by providing ways to explore hidden "treasures" inside them, the value of their collection will drastically increase.

Also, notwithstanding the "digital revolution" had many advantages, we can point out some negative effects. Users own huge music collections that need proper storage and labelling. Searching inside digital collections arise new methods for accessing and retrieving data. But, sometimes there is no metadata —or only the file name— that informs about the content of the audio, and that is not enough for an effective utilization and navigation of the music collection. Thus,

users can get lost searching into the digital pile of his music collection. Yet, nowadays, the web is increasingly becoming the primary source of music titles in digital form. With millions of tracks available from thousands of websites, finding the *right* songs, and being informed of newly music releases is becoming a problematic task. Thus, web page filtering has become necessary for most web users.

Beside, on the digital music distribution front, there is a need to find ways of improving music retrieval effectiveness. Artist, title, and genre keywords might not be the only criteria to help music consumers finding music they like. This is currently mainly achieved using cultural or editorial metadata ("this artist is somehow related with that one") or exploiting existing purchasing behaviour data ("since you bought this artist, you might also want to buy this one"). A largely unexplored (and potentially interesting) complement is using semantic descriptors automatically extracted from the music audio files. These descriptors can be applied, for example, to recommend new music, or generate playlists.

1.4 Motivation

In the past twenty years, the signal processing and computer music communities have developed a wealth of techniques and technologies to describe audio and music contents at the lowest (or close-to-signal) level of representation. However, the gap between these low-level descriptors and the concepts that music listeners use to relate with music collections (the so-called "semantic gap") is still, to a large extent, waiting to be bridged. Although there is a lot of ongoing work to bridge that semantic gap in several disciplines —image retrieval, music information retrieval, etc.— there not exist applications, for the real end-user, that exploits the links and relationships among low and mid-level descriptions, and human understanding.

During the last years, music recommendation paradigms have been largely studied, and lots of new systems have appeared with more or less success. Yet, most of the approaches applied the existing classic methods of recommender systems into the music domain. The music recommendation, as a multi-faceted approach and as an example to overcome the existing semantic gap between users and the content, has not yet been addressed by the researchers. However, it is not only a very interesting and challenging topic, but it also allows for very

practical and relevant applications in music information retrieval.

1.5 Goals and Contributions

The main goal of this Thesis is to study the music recommendation problem. To achieve this goal, we introduce some research open questions surrounding this problem. It is the purpose of this Thesis to answer these questions:

- Which are the existing approaches in music recommendation, both on the literature and commercial world? How much has been their success?
- How can different approaches be mixed in a single system?
- How can the preferences of a user be modelled and exploited for music recommendation?
- Is the inclusion of music description feasible, useful or advanteageous for music recommendation?
- How music recommendation systems can be evaluated?

We present here a rough overview of the contributions of this dissertation, which are related to the hypothesis that we want to verify. In this Thesis we:

1. Review current efforts in music recommendation. This multidisciplinary study ranges from audio processing methods to music cognition approaches, and social networks. We also study how the current literature related to recommendation can be applied to the music (this review is presented in chapters 2 and 3).
2. Study how to establish a relationship between mid–level description from audio analysis, and user preferences. This study comprises a multi–facet description of the music assets, including editorial, cultural and acoustic metadata (this study is introduced in chapters 4 and 5).
3. Implement a set of prototypes that examine different approaches of music recommendation (this contribution is presented in chapters 6 and 7).

Some remaining tasks, such as the evaluation of music recommendations, are detailed on the conclusions and future work (chapter 8), where we outline the road map of the planned future work research, and look at the tasks that need to be covered to finalize the PhD thesis.

1.6 Thesis Outline

This Thesis is structured as follows: chapter 2 introduces to the reader the basics of the recommendation problem. After that, chapter 3 adapts the general problem of recommendation to the music domain, and presents the related work in this area. To characterize the music recommendation problem, it is needed to describe the assets related with the music field, and how to handle and represent these descriptions for a further recommendation process. This is covered in chapters 4 and 5, respectively. After that, chapters 6 and 7 present two prototypes that we have implemented. These prototypes demonstrate how to exploit music related content available on the web. Finally, chapter 8 draws some conclusions and discusses open issues and future lines for the PhD final Thesis.

Chapter 2

The recommendation problem

Recommendations are a part of everyday life. We usually rely on external knowledge and judgements to make informed decisions about a particular action. There are many factors which may influence the decision making. Ideally, a recommendation system should be able to track as many factors as possible.

Recommender systems have been built for entertainment content domains, such as: movie, music, books recommendation etc. [HKTR04]. Generally speaking, the reason people could be interested in using a recommender system is that they have so many items to choose from —in a limited period of time—, that they cannot evaluate all the possible options. A recommender should be able to bring and filter all this information to the user.

This chapter is structured as follows: the next section introduces a formal definition of the recommendation problem. After that, section 2.2 presents some use cases to stress the possible usages of a recommender. One of the most important issues for a recommender system is how to model users' preferences and their profiles. This is discussed in section 2.4, where the representation and exploitation of users' profiles is presented.

2.1 Formalizing the recommendation problem

Intuitively, the recommendation problem can be split into two subproblems. The first one is a prediction problem, and is about the estimation of ratings for a set of items that a user has not seen. The second problem is to recommend a list of N items —assuming that the system can predict ratings for yet unrated items.

Actually, the most relevant problem is the estimation. Once the system can estimate items into a totally ordered set, the recommendation problem is *as simple as* listing the first N items with the highest estimated value.

- The **prediction problem** can be formalised as follows [SKKR01]: Let $U = \{u_1, u_2, \dots u_m\}$ be the set of all users, and let $I = \{i_1, i_2, \dots i_n\}$ be the set of all possible items that can be recommended.

  Each user u_i has a list of items I_{u_i}. This list represents the items that the user has expressed his interests. Note that $I_{u_i} \subseteq I$, and it is possible that I_{u_i} be empty[1], $I_{u_i} = \emptyset$. Then, the function, $P_{a,j}$ is the predicted likeliness of item i_j for the active user u_a, such as $i_j \notin I_{u_i}$.

- The **recommendation problem** is reduced to bringing a list of N items, $I_r \subset I$, that the user will like the most (i.e the ones with higher $P_{a,j}$ value). The recommended list cannot contain items from the user's interests, i.e. $I_r \cap I_{u_i} = \emptyset$.

The space I of possible items can be very large. Similarly, the user space U, can also be enormous. In recommender systems, the prediction function is usually represented by a rating. User ratings are triplets (u, i, r) where r is the value assigned —explicit or implicitly— by the user to a particular item. Usually, this value is a real number (e.g from 0 to 1), or a value in a discrete range (e.g from 1 to 5), or a binary variable (e.g *like*/*dislike*).

There are many approaches to solve the recommendation problem. One approach is that the user gives feedback to the system, so the system can provide informed guesses, based on ratings that other users have provided. This approximation is called *collaborative filtering*. Another approach is that the system collects information describing the items and then, based on the user's preferences, the system is able to predict which items the user will like. This approach is known as *content–based filtering*, as it does not rely on other users' ratings but on the description of the items. Another approach is *demographic filtering*, that stereotypes the kind of users that like a certain item. Finally, *context–based filtering* uses contextual information to describe the objects. Yet, there is the method that combines some of the previous approaches, named the *hybrid* approach. Section 2.4.2 presents all these approaches. Before presenting

[1]Specially when the user creates an account to a recommender system

the methods to solve the recommendation problem, the next section explains the most common usages of a recommender. After that, section 2.4 explains how to model users' preferences and exploit their profiles (by using some of the before mentioned methods).

2.2 Use Cases

Once the problem has been specified, the next step is to define general use cases that makes a recommender system useful. In [HKTR04], Herlocker et al. identify some usages of a recommender:

- **Find good items**. The aim of this use case is to provide a ranked list of items, along with a prediction of how much the user would like each item. Ideally, a user would expect some novel items that he has not seen before.
- **Find all good items**. The difference of this use case from the previous one is with regard the coverage. In this case, the user wants to be assured that false positive rate is low, thus the precision is high.
- **Recommend sequence**. This use case aims at bringing to the user an ordered sequence of items that is pleasing as a whole. A paradigmatic example is the playlist generation of a music recommender.
- **Just browsing**. In this case, users find pleasant to browse into the system, even if they are not willing to purchase any item. Simply as an entertainment.
- **Find credible recommender**. Users do not automatically trust a recommender. Then, they "play around" with the system to see if the recommender does the job well. A user interacting with a music recommender will probably search for one of his favourite artists, and check —even evaluate— the output results (e.g. similar artists, playlists generation, etc.)
- **Improve profile**. Users contribute with ratings[2] because they believe that are improving their own profiles, as well as the recommendations they will receive. This use case (and the next ones) is important in recommenders that have a strong community component.

[2]These ratings do not necessarly mean explicit item ratings, it can be implicit feedback.

- **Express self**. For some users is important to express their opinions. A recommender that offers a way to communicate and interact with other users (via forums, weblogs, etc.) allows the self–expression of users.

- **Help others**. There are users that feel happy to contribute with ratings. They believe that the whole community benefits from their contributions, and some people will discover new items due to their work.

- **Influence others**. This use case is the most negative of the ones presented. There are some situations where users might want to influence the community in viewing or purchasing a particular item. E.g: Movie studios can rate high their latest new release, to push others to go and see the movie. In a similar way, record labels can try to promote artists into the recommender.

These use cases' characterization is important when evaluating a recommender. The first task of the evaluators should be to identify the most important use cases for which the recommender will be used.

2.3 Factors affecting the recommendation problem

There are some inherent features of the data sets (i.e users and items) involved in the recommendation problem. **Data sparsity** and **high dimensionality** are two of the most common properties of the data sets. These make more difficult the recommendation problem. The **bias** of the items data collection is another important issue. E.g. does a music recommender deal only with *popular* music, and it is excluding classical music?

Novelty detection is a very important feature for the recommendation problem. A recommender with low novelty rate only recommends obvious items to users. Obvious recommendations have two practical disavantages: for instance, users who are interested in those items could probably already know them, and secondly, managers in stores (i.e experts of the items' domain) do not need any recommender to tell them which products are popular overall. Although, obvious recommendations do have value for new users. Users like receiving some recommendations of items they already are familiar with [SS01]. This is related with

the *Find credible recommender* use case (see previous section, 2.2). Yet, there is a trade–off between the desire for novelty and the desire for high quality of the recommendations. A high novelty rate might mean, for a user, that the quality of the recommendation is poor, because the user wil not be able to identify most of the items in the list of recommendations.

Another important feature is the **serendipity**. That is the good luck in making unexpected and fortunate discoveries. A recommender should help the user to find a surprisingly interesting item that he might not be able to discover otherwise. Recommendations that are serendipitous are also novel.

The **coverage** of the recommender is another feature to take into account. The coverage of a recommender is a measure of the domain of items over which the system can form predictions or make recommendations. Low coverage of the domain might be less valuable to users, as it limits the space of possible items to recommend. Moreover, this feature is important for the *Find all good items* use case (see section 2.2). Low coverage and a big bias in the items data collection can be very frustrating for the users.

The **learning rate** curve of the recommender is another issue (also known as the cold–start problem [DE95], or the bottleneck problem). The learning rate is non–linear but asymptotic. This could be a problem when users start playing around with the system.

The **transparency** of the recommendations is another element to look at. A recommender should be able to explain to the user why the system recommend that items. This feature is also important for the *Find credible recommender* use case.

Modelling **user preferences**, including psychographic information is another challenging problem. Psychographic variables include attributes relating to personality, such as attitudes, interests, or lifestyles. It is not straightforward to encode all this information and make it useful for a system. This problem is similar to expressing user's needs via a query, in Information Retrieval (IR) systems. There is always a loss of information when formulating the query using a language that the machine can understand and process.

Finally, the **timestamp** of items could be an important factor of the recommender. The prediction function should take into account the *age* of the items. A common approach is to treat the older items as less relevant than the new ones

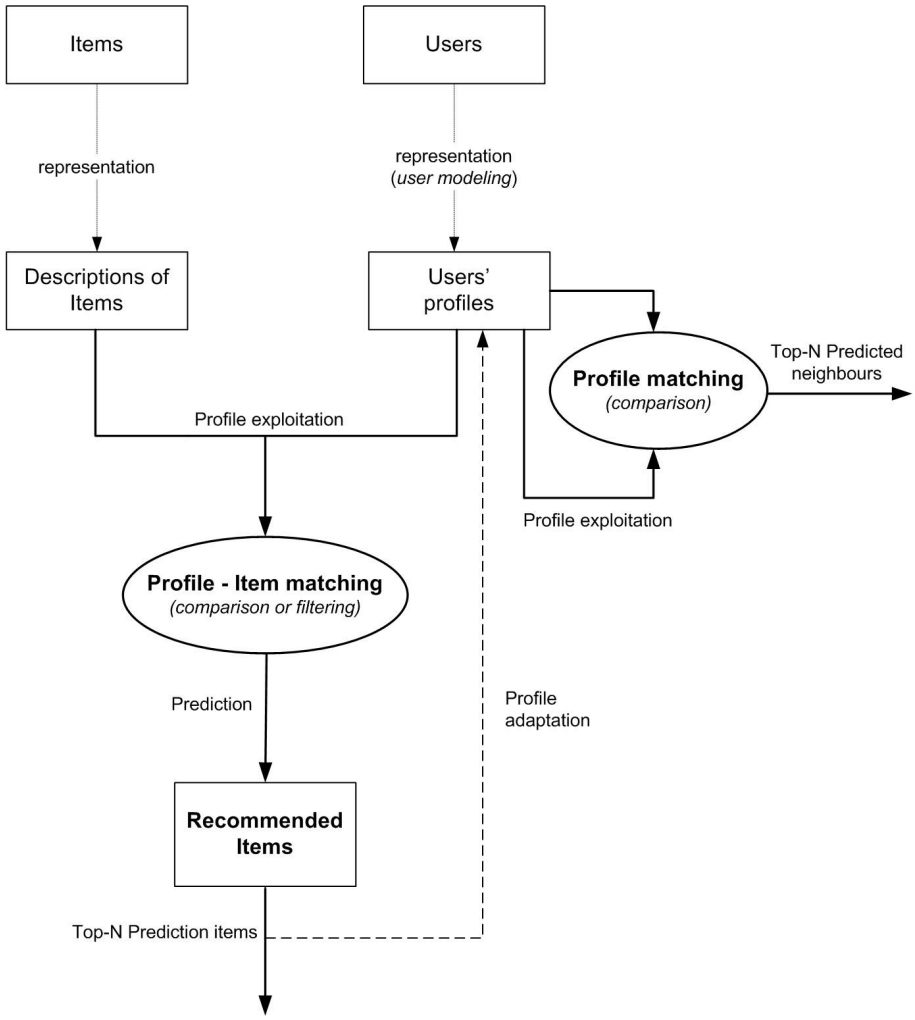

Figure 2.1: General model of the recommendation problem.

(thus, incrementing the novelty rate).

It is worth noting that most of these factors are dependant on the user, and are very difficult to evaluate with accurate metrics.

2.4 User profiling

The main elements of a recommender are: users and data items. Users need to be modeled in a way that the recommender can exploit their profiles and preferences. On the other hand, accurate descriptions of the items in the database is crucial to achieve good recommendations.

There are two key elements when describing user preferences: the generation

and maintenance of the profiles, and the exploitation of the profile [MLdlR03]. Profile generation involves the representation, initial generation, and adaptation techniques. Profile exploitation involves the information filtering method (i.e the recommendation method), the matching of profiles and items, and the user profiles matching (i.e creation of neighbourhoods).

Figure 2.1 describes the major entities and processes involved in the recommendation problem. The next sections explain the key elements of the diagram.

2.4.1 User profile representation

There are several approaches to represent user preferences. History of purchases in a e–Commerce website, the web navigation (analysis of links and time spent in each webpage), the listening habits (songs that a user listens to), etc. can be useful to represent users' profiles.

User profile initial generation

Yet, an important issue is the creation of the user's profile. The easiest way is to create an empty profile, that will be updated as soon as the user interacts with the system. Another approach is to manually create a profile. In this case, a system might ask users to register their interests (via keywords or topics) as well as some demographic informaton (e.g age, marital status, gender, etc.), geographic data (city, country, etc.) and psychographic data (interests, lifestyle, etc.). The main drawback is the user's effort, and the fact that maybe some interests could be still unknown by the user himself.

Another method to gather information is using a training set. The process is to ask the user to rate concrete items as relevant or irrelevant to his interests. The main problem, though, is to select representative examples. For instance, in the music domain, the system might ask for concrete genres or styles, and filter a set of artists to be rated by the user.

Finally, the last method to gather initial information is named stereotyping. This method is similar to a clustering problem, and the idea is to assign a user into a cluster of similar users that are represented by their stereotype. Nevertheless, it is worth to mention that in some cases it might not be necessary to generate an initial user's profile. For instance, recommenders that aquire user information

from a database (purchasing history), or community–based systems[3] —that keep track of a matrix with user–item ratings as profile— start with an empty profile.

User profile maintenance

Once the profile has been created, this does not remain static. Therefore, interests might (and probably will) change over time. Recommender needs up-to-date information to automatically update the user's profile. This is called relevance feedback. Users' feedback can be positive or negative. Usually, there is more feedack with regard to positive examples, although it has been proved that negative examples can improve the system.

Feedback can be explicit or implicit. Explicit feedback can come from ratings (in a discrete scale, e.g. from 0 to N, or binary such as $like/dislike$), though, it is proved that sometimes users rate inconsistenly [HSRF95]. Another way is getting explicit feedback from text comments. In this case, the sytem gathers users' comments about single items and present it to the target user, along with the recommendations. This extra information eases the decision–making process of the user, although the user has to process and interpret the text comments.

On the other hand, a recommender can get implicit feedback from the user. A system can infer the user's preferences passively by monitoring user's actions. For instance, by analyzing the history of purchases, the time spent on a webpage, the links followed by the user, or even analysing the media player (tracking the *play*, *pause*, *skip* and *stop* buttons).

User profile adaptation

As explained in the previous section, relevance feedback implies that the system has to adapt to the changes of users' profiles. The techniques to adapt to the new interests and forget the old ones can be done in three different ways. First, done manually by the user, although this requires some effort to the user. Secondly, by adding new information into the user's profiles, but then the old interests are never forgotten. And third, by gradually forgetting the old interests and promoting the new ones [WK96].

[3]Also known as Collaborative filtering systems

2.4.2 User profile exploitation

Once the user profile has been created, and the system is able to adapt it to the new user's interests, the next step is to exploit user preferences to recommend interesting items.

The profile exploitation is tightly related with the information filtering method. In this case, the process of filtering information can be classified according to the similarity among users (user profile matching), and the similarity among user's profile and items (user profile–item matching).

The method adopted for filtering the information has led to the standard classification of recommender systems, that is: demographic filtering, collaborative filtering, content–based and hybrid approaches [MLdlR03]. We add another method to the classification, that is the context–based, which has grown popularity recently due to the feasability of getting information *about* the items (e.g gathering information from weblogs, reviews about the item, etc.).

Demographic filtering

Demographic filtering can be used to identify the kind of users that like a certain item [Ric79]. For example, one might expect to learn the type of person that likes a certain singer (e.g try to find the stereotypical user that listens to *Hilary Duff*[4] singer). This technique, then, classifies users' profiles in clusters according to some personal data (age, marital status, gender, etc.), geographic data (city, country) and psychographic data (interests, lifestyle, etc.).

The main problems of this method is that a system recommends the same items to people with similar demographic profiles, so recommendations are too general (or, at least, not very specific for a given user). Another drawback is the generation of the profile, that needs some effort from the user. Some approaches try to get (unstructured) information from user's webpages, weblogs, etc. In this case, text classification techniques are used to create the clusters, and classify the users [Paz99]. All in all, this is the simpliest method to exploit the user's profile.

An early example of a demographic system is Grundy [Ric79]. Grundy recommended books based on personal information gathered from an interactive

[4]http://www.hilaryduff.com

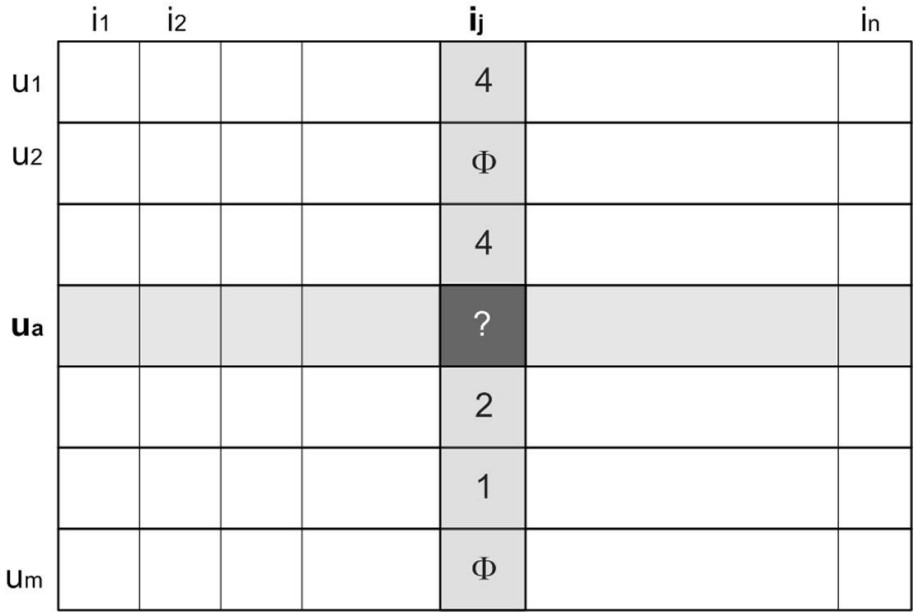

Figure 2.2: User–item matrix for the collaborative filtering approach. The predicted rating value of item i_j, for the active user u_a can be computed as the mean of the ratings' values of users similar to u_a.

dialogue.

Collaborative filtering

The main idea of collaborative filtering (CF) is that the user gives feedback to the system, so the system can provide informed guesses, based on ratings that other users have provided. The first system that implemented the collaborative filtering method, in 1992, was the Tapestry project at Xerox PARC [GDOT92]. The project coined the *collaborative filtering* term. Other early systems are: a music recommender named Ringo ([Sha94], [SM95]), and GroupLens, a system for rating USENET articles [RIS+94]. A compilation of other systems from that time period can be found at [RV97].

CF methods work by building a matrix of users' preferences (or ratings) for items. Each row represents a user profile, whereas the columns are items. The value $R_{u,i}$ is the rating of the user u_i for the item i_j. Figure 2.2 depicts the matrix of user–item ratings. The predicted rating value of item i_j, for the active user u_a, $P_{a,j}$, can be computed as the mean of the ratings' values of users similar to u_a. This technique allows to solve the user profile–item matching problem.

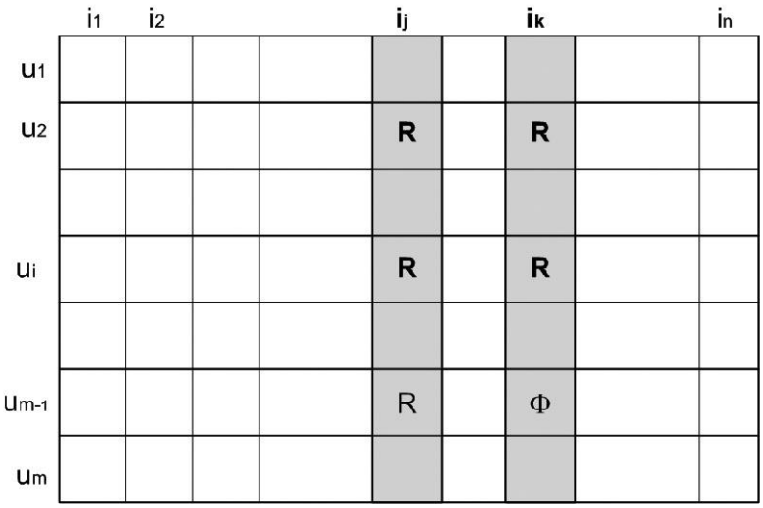

Figure 2.3: User–item matrix with co–rated items for item–based similarity. To compute the similarity between items i_j and i_k, only users u_2 and u_i are taken into account, but u_{m-1} is not because it has not rated both items (i_k rating value is $\emptyset$).

Equation 2.1 shows the predicted rating score of item i_j, for user u_a. $\bar{R}_a$ is the average rating of user u_a.

$$P_{a,j} = \bar{R}_a + \sum_{u \in Neighbours(u_a)} sim(u_a, u)(R_{u,j} - \bar{R}_u) \tag{2.1}$$

This approach is also known as *Memory–based* collaborative filtering algorithms. Yet, to predict $P_{a,j}$, the algorithm needs to know beforehand the set of users similar (i.e neighbours) to the active user, u_a, and how similar they are ($sim(u_a, u)$). This is analogous to solve the user profile matching problem (see figure 2.1). The most common approaches to find the neighbours of the active user are cosine similarity (see equation 2.2), *K–Nearest Neighbours* and clustering based on stereotypes [MLdlR03].

On the other hand, the *Model–based* (or item–based) method is based on item similarity. This method looks into the set of items a user has rated, and computes the similarity among the target item (to decide whether is worth to recommend it to the user or not). Figure 2.3 depicts the co–rated items from different users. In this case it shows the similarity between items i_j and i_k. Note that only users u_2 and u_i are taken into account, but u_{m-1} is not because it has not rated both items. The first step is to obtain the similarity between the two items i_j and i_k.

This can be achieved by using cosine similarity, correlation–based similarity, or adjusted cosine similarity methods [SKKR01].

Let the set of users who rated i and j be denoted by U, and $R_{u,i}$ denotes the rating of user u on item i. Equation 2.2 shows the definition of the cosine similarity:

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|i\| * \|j\|} = \frac{\sum_{u\in U} R_{u,i}R_{u,j}}{\sqrt{\sum_{u\in U} R_{u,i}^2}\sqrt{\sum_{u\in U} R_{u,j}^2}} \tag{2.2}$$

However, for the item–based similarity, the cosine similarity does not take into account the differences in rating scale between different users. The adjusted cosine similarity (equation 2.3) makes use of user average rating from each co–rated pair, and copes with the limitation of cosine similarity. $\bar{R}_u$ is the average rating of the u–th user:

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u\in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u\in U}(R_{u,j} - \bar{R}_u)^2}} \tag{2.3}$$

Correlation–based similarity commonly uses the $Pearson-r$ correlation. The correlation between two variables reflects the degree to which the variables are related. Equation 2.4 defines the correlation similarity. $\bar{R}_i$ is the average rating of the i–th item:

$$sim(i,j) = \frac{Cov(i,j)}{\sigma_i\sigma_j} = \frac{\sum_{u\in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u\in U}(R_{u,j} - \bar{R}_j)^2}} \tag{2.4}$$

Once the similarity among items has been computed, the next step is to predict to the target user a value for the active item. A common way is to capture how the user rates items similar to the active item. This can be done by computing the sum of the user's ratings —only for the items similar to the active item— weigthed by the item similarity.

Collaborative filtering is one of the most used methods of existing recommender systems, yet the approach presents some drawbacks:

- **Data sparsity** and **high dimensionality** are inherent properties of the datasets (see section 2.3). With a relative small number of users and a large item dataset collection, the main problem is the low coverage of the

users' rating among the items. Also, it becomes difficult to find reliable neighbours as well as recommending good items.

- Another problem (related with the previous one) is that users with ***atypical* tastes** —i.e that vary from the norm— will not have many users as neighbours. Thus, this will lead to poor recommendations. This problem is also known as **gray sheep** [CGMM99].

- New items appearing in the database cannot be recommended until users start rating it. This issue is known as the **early–rater** problem [AZ97]. Moreover, the first user that rates new items gets only little benefit (a new item does not match with any other item).

- **Cold–start problem** This problem appears for both elements of music recommenders: users and items. Due to CF is based on users' ratings, new users with only a few ratings become more difficult to categorize. The same problem occurs with new items, because they do not have many ratings when being added to the item collection.

- The fact that the approach is based only on users ratings, but it does not take into account the description of the items, implies that is a subjective method. So, the approach is content–agnostic.

- The **popularity bias** is another interesting problem that happens in this method. It is analogous to the *"rich gets richer"* paradigm. Popular items of the data set are similar to (or related with) lots of items. Thus, it is more probable that the system recommends these popular items. This behaviour can be modeled using the graph theory applied to complex networks[5].

Content–based filtering

In the content–based (CB) filtering approach, the recommender collects information describing the items and then, based on the user's preferences, it predicts which items the user will like. This approach does not rely on other users' ratings but on the description of the items. The process of characterizing the item data

[5]In [CCKMB06], we have shown that collaborative filtering —to recommend artists in the music domain— present the topology of a *scale free* network. This motivates that the recommendations are biased toward popular items.

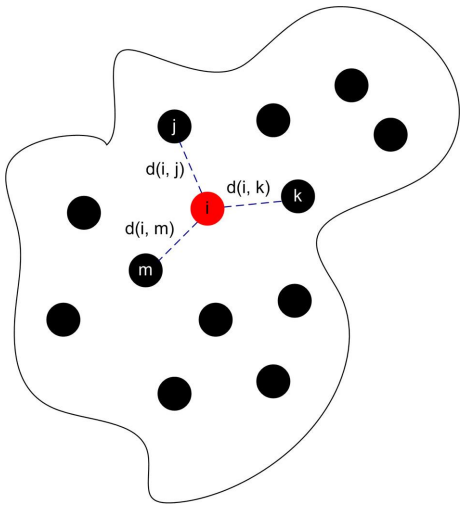

Figure 2.4: Content–based similarity distance among items.

set can be either automatic —e.g extracting features by analyzing the content—, or based on manual annotations made by the domain experts.

CB approaches have its roots in the information retrieval (IR) field. The early systems focused on text domain, and applied techniques from IR to extract *meaningful* information from the text. Yet, recently have appeared some solutions that cope with more complex domains, such as music recommendation. This has been possible, partly, because the multimedia community emphasized on and improved the feature extraction algorithms. The key component of this approach is the similarity function among items (see figure 2.4).

Similarity functions are mostly based on computing the distance between two items. Although, similarity measures are not always objective. In some domains, similarity is very context–dependant, and subjectivity is a big factor of the measure. Nevertheless, content–based similarity usually only takes into account an objective distance among items, without introducing the subjective factor into the metric. Most of the distance metrics deals with numeric attributes. Some common distances are: euclidean (equation 2.5), Manhattan (equation 2.6), chebychev (equation 2.7), cosine distance for vectors (see previously defined equation 2.2), and Mahalanobis distance (equation 2.8).

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.5}$$

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.6}$$

$$d(x,y) = max_{i=1..n}|x_i - y_i| \tag{2.7}$$

All the previous distances (euclidean, Manhattan and chebychev) are assuming that the attributes are orthogonal. The Mahalanobis distance is more robust to the dependencies among attributes, as it uses the covariance matrix S:

$$d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)} \tag{2.8}$$

If the attributes are nominal (not numeric), a delta function can be used. A simple definition of a delta function could be: $\delta(a,b) = 0 \Leftrightarrow a = b$, and $\delta(a,b) = 1$ otherwise. Then, a distance metric among nominal attributes can be defined as (where ω is a reduction factor, e.g $\frac{1}{n}$):

$$d(x,y) = \omega \sum_{i=1}^{n} \delta(x_i, y_i) \tag{2.9}$$

Finally, if the distance to be computed has to cope with both numeric and nominal attributes, then the final distance has to combine two equations (2.9 for nominal attributes and one of 2.5. . . 2.8 for numeric attributes).

It is worth noting that all CB approaches have some drawbacks:

- The cold–start problem occurs when a new user enters to the system. The system has to *learn* from user's preferences.
- Another caveat could be the novelty problem. Assuming that the similarity function works accurately, then one might assume that a user will always receive items *too* similar to the ones in his profile. To cope with this shortcoming, the recommender could add some randomness to the prediction function to spread out the *eclecticness* of the recommended items.
- Depending on the complexity of the items' domain, another drawback is the limitation of the features that can be (automatically) extracted from the objects. For instance, in the multimedia arena, nowadays, is very difficult to extract high–level descriptors with a clear meaning for the user. Music analysis is not ready yet to correctly extract *moods* but, on the other

hand, it does the job well when dealing with descriptors such as: harmony, rhythm, etc. Thus, an item description is not close enough to the user, but still that description allows to compute similarity among items (e.g songs).

- Another shortcoming is that the recommender is focused on finding similarity among items, using only the features describing the items. The method is limited by the features that are explicitly associated with the items. This means that subjectivity (or personal opinions) is not taken into account when recommending items to users.

Moreover, CB methods solve some of the shortcomings of the collaborative filtering. The early–rater problem disappears. When adding a new item into the collection —and computing the similarity among the rest of the items— it can be recommended without being rated by a user. The popularity bias is solved too. As there is no human intervention in the process, all the items are considered to be of equal importance.

Context-based filtering

Context is any information that can be used to characterize the situation of an entity. Context–based filtering uses contextual information to describe and characterize the objects. This method considers not only users and items datasets, but external information that refers to them. To compare content and context–based filtering, a clear example is the different methods used for spam detection. The common one is based on the text analysis of the mail, whereas context filtering does not deal with the content of the mail. It rather uses the context of the SMTP connection to decide whether a mail will be marked as spam or not.

In case that the context information about the item is in textual form, classic measures of Information Retrieval and Text Mining can be applied to characterize the item. For instance vector space–based models —using tf/idf as the weighting function—, can be used to model both items and users' profiles. Similarity between an item and user's profile can be computed using cosine–based similarity (see section 2.4.2). Regarding tf/idf, tf stands for Term Frequency, whereas idf is the Inverse Document Frequency [SM86]. The term frequency in a given document measures the importance of the term t_i within that particular document. Equation 2.10 defines tf:

$$tf = \frac{n_i}{\sum_k n_k} \tag{2.10}$$

with n_i being the number of occurrences of the considered term, and the denominator is the number of occurrences of all the terms in the document.

The Inverse Document Frequency, idf, measures the general importance of the term, in the whole collection of items:

$$idf = \log \frac{|D|}{|(d_i \supset t_i)|} \tag{2.11}$$

where $|D|$ is the total number of items, and the denominator counts the number of items where t_i appears. Finally, the weighting function $w_{t,j}$, of a term t in the item d_j is computed as:

$$w_{t,j} = tf \cdot idf \tag{2.12}$$

Cosine–based similarity between an item i, and a user profile u_j is defined as:

$$sim(i,j) = \frac{\sum_t w_{t,i} w_{t,j}}{\sqrt{\sum_t w_{t,i}^2}\sqrt{\sum_t w_{t,j}^2}} \tag{2.13}$$

In the music domain, context information surrounding the artists and/or songs (i.e the items to recommend) could come from editorial data —album reviews, artist biographies, artist relationships, etc— as well as all sort of users' opinions grabbed from their weblogs, for instance.

Hybrid methods

The main purpose of an hybrid method is to achieve a better prediction by combining some of the previous stand–alone approaches. Most commonly, collaborative filtering is combined with other techniques

There are different methods to integrate several techniques into a hybrid recommender. In [Bur02], Burke defines several approaches:

- **Weighted**. A hybrid method that combines the ouput of separate approaches using, for instance, a linear combination of the scores of each recommendation technique.

- **Switching**. The system uses some criterion to switch between recommendation techniques. One possible solution is that the system uses a technique, and if the results are not confident enough, it switches to another technique to improve the recommendation process.

- **Mixed**. In this approach, the recommender does not combine but expand the description of the data sets by taking into account both users' ratings and the description of the items. The new prediction function has to cope with both types of descriptions, and present it to the user.

- **Cascade**. The cascade involves a step by step process. In this case, a recommendation technique is applied first, producing a coarse ranking of items. Then, a second technique refines the obtained results.

A hybrid method can alleviate some of the drawbacks that suffer a single technique.

2.5 Summary and Conclusions

This chapter has presented and formalised the recommendation problem. The main components of a recommender are users and items. Based on user preferences and user profile exploitation, a recommender can solve the problem of presenting interesting items to the users. Table 2.1 summarizes all the elements involved in the recommendation problem, that is: the user profiling generation, adaptation and exploitation.

The next chapter 3 applies all the concepts presented here into a specific domain, that is music. Moreover, the special requirements to solve the music recommendation problem are outlined too.

User profile generation	
Initial generation	*empty*
	manual
	training set
	stereotyping
Maintenance	*implicit relevance feedback*
	explicit relevance feedback
User profile adaptation	*manual*
	add new information
	gradually forget old interests
User profile exploitation	
Filtering method	*demographic*
	collaborative filtering
	content − based filtering
	context − based filtering
	hybrid methods
Matching	*user profile − item*
	user profiles (neighbours)

Table 2.1: Summary of the elements involved in the recommendation problem.

Chapter 3

Music recommendation

This chapter introduces to the reader the music recommendation problem. First, it presents the general recommendation problem adapted to the music domain. After that, section 3.2, discusses the user modelling and how to link it with the music concepts. User exploitation (via collaborative filtering, content and context–based methods) is presented in section 3.2.2. The section focus on the related work and existing approaches to get music recommendations, mostly based on artist (and song) similarity. After that, we present a paradigmatic use case, that is playlist generation, based on different approaches. Finally, a brief overview of some relevant systems is outlined in section 3.3.

3.1 Recommendation task

The main task of a music recommendation system is to propose, to the end-user, interesting and unknown artists —and their available tracks, if possible—, based on user's musical taste. But musical taste and music preferences are affected by several factors, including demographic and personality traits. The combination of music preferences and personal aspects —such as: age, gender, origin, occupation, musical education, etc.— could improve music recommendations [UvS02].

Most of the work done in music recommendation has been focused on presenting a list of artists, or creating an ordered sequence of songs (i.e a playlist) that the user might like. To achieve this, the most common approaches have been to exploit users' profiles via collaborative filtering or content–based filtering. Although, we foresee that music recommendation involves a broader experience

with the user, more than presenting a list of recommended artists or generating playlists. In this sense, there is a lot music related information on Internet: music performed by "unknown" artists that can suit perfectly for new recommendations, new music releases of mainstream artists, related news, announcements of concerts, album reviews, podcasts sessions, users' opinions on their weblogs, etc. Nowadays, music sites start to syndicate their web content —noticing the user about new releases, artist's related news, upcoming gigs, etc.— mostly in the form of RSS feeds. The RSS abbreviation is variously used to refer to the following standards: Really Simple Syndication (RSS 2.0), Rich Site Summary (RSS 0.91 and 1.0) or RDF Site Summary (1.0). For instance, iTunes Music Store[1] provides an RSS feed generator[2], updated once a week, that publishes the new releases. A music recommendation system should take advantage of these publishing services, as well as integrating them into the system, in order to filter and recommend music related information to the user.

3.2 User profiling

The process of recommending music is highly dependant on the user. User modelling process is a crucial step to *understand* user preferences. However, in the music field, there have been a few attempts to explicitly extend user profiles by adding music related information. Next section presents the more revelant approaches.

3.2.1 User profile representation

As mentioned in section 2.4, user modelling has been studied for many years. On the other hand, extending users' profiles with music related information has not been largely investigated. Although, it is an interesting way to communicate with other people, and to express music preferences[3]. Music is an important vehicle for telling other people something relevant about our personality, history, etc. The music information to be added in the profile should be related with user's interests and user's habits.

[1] http://www.apple.com/itunes

[2] http://phobos.apple.com/WebObjects/MZSearch.woa/wo/0.1

[3] Nowadays, it is very common to embed to a webpage a small plugin that displays the most recent tracks a user has played.

The most relevant proposals are: the User modelling for Information Retrieval Language, the related description schemas defined by the MPEG-7 standard, and the Friend of a Friend (FOAF) initiative —hosted by the Semantic Web community. The complexity in terms of semantics is increasing in each of the proposals. The next sections present these approaches.

User modelling for Information Retrieval

The User modelling for Information Retrieval Language (UMIRL), proposed by Chai and Vercoe [CV00], allows to describe perceptual (qualitative) features of the music. It is specially designed for music information retrieval systems. The profile can contain both demographic information and direct information about the music objects: favourite bands, styles, songs, etc. Moreover, a user can add his definition of a perceptual feature, and his meaning, using music descriptions. For instance: "a *romantic piece* has a slow tempo, lyrics are related with *love*, and has a soft intensity". And the context to use this feature is while having a special dinner with user's girlfriend.

The representation they proposed uses the XML syntax, without any schema or document type definition to validate the profiles. Example 3.1, shows a possible user profile:

```
<user>
  <generalbackground>
    <name>Joan Blanc</name>
    <education>MS</education>
    <citizen>Catalan</citizen>
    <sex>male</sex>
  </generalbackground>
  <musicbackground>
    <education>none</education>
    <instrument>guitar</instrument>
    <instrument>vocal</instrument>
  </musicbackground>
  <musicpreferences>
    <genre>blues</genre>
    <genre>rock</genre>
    <composer>Johann Sebastian Bach</composer>
    <artist>The Dogs Amour</artist>
    <sample>
      <title>Two hearts beat as one</title>
      <artist>U2</artist>
    </sample>
  </musicpreferences>
  <habit>
```

```
    <context>Happy
      <tempo>very fast</tempo>
      <genre>rock</genre>
    </context>
    <perceptualfeature>Romantic
      <tempo>very slow</tempo>
      <intensity>soft</intensity>
      <lyrics>*love*</lyrics>
    </perceptualfeature>
    <context>Dinner with fiance
      <perceptualfeature>Romantic</perceptualfeature>
    </context>
  </habit>
</user>
```

Listing 3.1: Example of a user profile in UMIRL.

This proposal is interesting, and one of the first attempts in the Music Information Retrieval community. The main goal was to propose a representation format, as a way to interchange profiles among systems. Though, it lacks formal semantics to describe the meaning of their descriptors and attributes.

User Preferences in MPEG-7

MPEG-7, formally named Multimedia Content Description Interface, is an ISO/IEC standard developed by the Moving Picture Experts Group (MPEG) [MS02]. The main goal of the MPEG-7 standard is to provide structural and semantic description mechanisms for multimedia content. A more detailed explanation of the standard is presented in section 5.2. The standard provides a set of description schemes (DS) to describe multimedia assets. At this moment, we only focus on the descriptors that allow to describe user preferences of multimedia content, while a concise description of the whole standard will be presented later on (in section 5.2).

User preferences in MPEG-7 include content filtering, searching and browsing preferences. The usage history, which represents the user history of interaction with multimedia items, can be denoted too. Filtering and searching preferences include the user preferences regarding classification (i.e country of origin, language, available reviews and ratings, reviewers, etc.) and creation preferences. The creation preferences describes the creators of the content (e.g. favourite singer, guitar player, composer, and music bands). Also, it allows to define a set of keywords, location and a period of time. Using a preference value attribute, the user can express positive (likes) and negative (dislikes) preference for each

descriptor. Next example shows a hypothetical user profile definition:

```
<UserPreferences>
  <UserIdentifier protected="true">
    <Name xml:lang="es">Joan Blanc</Name>
  </UserIdentifier>
  <FilteringAndSearchPreferences>
    <CreationPreferences>
      <Title preferenceValue="8">To bring you my love</Title>
      <Creator>
        <Role>
          <Name>Singer</Name>
        </Role>
        <Agent xsi:type="PersonType">
          <Name>
            <GivenName>Polly Jean</GivenName>
            <FamilyName>Harvey</FamilyName>
          </Name>
        </Agent>
      </Creator>
      <Keyword>dramatic</Keyword>
      <Keyword>fiery</Keyword>
      <DatePeriod>
        <TimePoint>1995-01-01</TimePoint>
        <Duration>P1825D</Duration>
      </DatePeriod>
    </CreationPreferences>
  </FilteringAndSearchPreferences>
</UserPreferences>
```

Listing 3.2: Example of a user profile in MPEG-7.

MPEG-7 usage history is defined in the usage history description scheme. *UsageHistory DS* contains a history of user actions. It contains a list of actions (play, play-stream, record, etc.), with an associated observation period. The action has a program identifier (an identifier of the multimedia content for which the action took place) and, optionally, a list of related links or resources.

In [TS05], Tsinaraki and Christodoulakis present a way to overcome with some of the limitations of describing user preferences in MPEG-7. They argue that there is still some lack of semantics when defining user preferences. For example, filtering and search preferences allow to specify a list of textual keywords, without being related with any taxonomy nor ontology. More complex descriptions are useful to determine user preferences. The semantic model (proposed in [TS05]) extends the creation preference with the semantic descriptors defined in the MPEG-7 standard. Finally, the implementation is integrated into a framework based on an upper ontology that covers the MPEG-7 multimedia description schemes. That upper ontology uses the OWL notation. OWL is the

Web Ontology Language used in the context of the Semantic Web. Section 5.3 presents the OWL language.

User profiling in the Semantic Web

The Semantic Web is an extension of the current Web that allows to the machines to find, share, and combine information more easily. It relies on machine-readable information and metadata expressed in the Resource Description Framework (RDF) language. An introduction about the Semantic Web, OWL and RDF is presented in section 5.3. In this section we only focus on how to model user preferences.

In the context of the Semantic Web, there is a clear interest to create a Web of machine-readable homepages describing people, the links between them and the things they like, create and do. The FOAF (Friend Of A Friend) project provides conventions and a language "to tell" a machine the sort of things a user says about himself in his homepage. FOAF is based on the RDF/XML vocabulary[4]. As we noted before, the knowledge hold by a community of "peers" about music is also a source of valuable metadata. FOAF nicely allows to relate and connect people.

FOAF profiles include demographic information (name, gender, age, sex, nickname, homepage, depiction, web accounts, etc.) geographic (city and country, geographic lattitude and longitude), social information (relationship with other persons), pyschographic (i.e user's interests) and behavioural (usage patterns). There are some approaches that allow modelling music taste in a FOAF profile. The simplest way is to show an interest for an artist:

```
<foaf:interest>
  rdf:resource="http://www.pjharvey.net"
  dc:title="P.J.Harvey" />
```

Listing 3.3: Example of a user interest using FOAF.

Already built–in within FOAF, there are ways to say that a user is interested in a topic. Even though there is no taxonomy of topics, this example gives more general information than the one shown in the previous example:

```
<foaf:topic_interest>
  <rdf:Description>
    <dc:subject>Alternative</dc:subject>
```

[4]http://www.w3.org/RDF/

```
    <dc:description>
      Rock music style from independent artists (...)
    </dc:description>
  </rdf:Description>
</foaf:topic_interest>
```

Listing 3.4: Example of a user topic interest using FOAF.

The Semantic Web approach facilitates the integration of different ontologies. Next example shows how to *embed* an album —using the *MusicBrainz* ontology— into a FOAF interest, as well as the rating of the album, stated by the user.

```
<foaf:interest>
  <mm:Album rdf:about="http://musicbrainz.org/album/24e5b7f5-14cd
     -4a65-b87f-91b5389a4e3a">
    <dc:title>To bring you my love</dc:title>
    <review:hasReview>
      <review:Review>
        <review:rating>8</review:rating>
        <dc:description>
          A classic album from P.J. Harvey (...)
        </dc:description>
      </review:Review>
    </review:hasReview>
    <dc:creator>
      <mm:Artist rdf:about="http://musicbrainz.org/artist/
         e795e03d-b5d5-4a5f-834d-162cfb308a2c"/>
    </dc:creator>
  </mm:Album>
</foaf:interest>
```

Listing 3.5: FOAF example of a song description that a user is interested in.

To conclude this section, the next example 3.6 shows a complete FOAF profile. We can see that the profile contains demographic and geographic information, as well as user's interests.

```
<rdf:RDF
  XML namespaces here >
 <foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
  <admin:generatorAgent
    rdf:resource="http://foafing-the-music.iua.upf.edu"/>
  <admin:errorReportsTo
    rdf:resource="mailto:ocelma@iua.upf.edu"/>
 </foaf:PersonalProfileDocument>
 <foaf:Person rdf:ID="me">
  <foaf:nick>ocelma</foaf:nick>
  <foaf:dateOfBirth>04-17</foaf:dateOfBirth>
  <foaf:gender>male</foaf:gender>
  <foaf:based_near geo:lat='41.401' geo:long='2.159' />
```

```
  <foaf:holdsAccount>
   <foaf:OnlineAccount>
    <foaf:accountName>ocelma</foaf:accountName>
    <foaf:accountServiceHomepage
      rdf:resource="http://www.blogger.com"/>
   </foaf:OnlineAccount>
  </foaf:holdsAccount>
  <foaf:holdsAccount>
   <foaf:OnlineAccount>
    <foaf:accountName>ocelma</foaf:accountName>
    <foaf:accountServiceHomepage
      rdf:resource="http://last.fm"/>
   </foaf:OnlineAccount>
  </foaf:holdsAccount>
  <foaf:mbox_sha1sum>ce24ca...a1f0</foaf:mbox_sha1sum>
  <foaf:interest>
   <foaf:Document rdf:about="http://www.gretsch.com">
    <dc:title>Gretsch guitars</dc:title>
   </foaf:Document>
  </foaf:interest>
  <foaf:interest>
   <foaf:Document
    rdf:about="http://www.tylaandthedogsdamour.com/">
    <dc:title>The Dogs Amour</dc:title>
   </foaf:Document>
  </foaf:interest>
  <foaf:interest>
    <mm:Album rdf:about="http://musicbrainz.org/album/24e5b7f5-14
       cd-4a65-b87f-91b5389a4e3a">
      <dc:title>To bring you my love</dc:title>
      <review:hasReview>
        <review:Review>
          <review:rating>8</review:rating>
          <dc:description>
            A classic album from P.J. Harvey (...)
          </dc:description>
        </review:Review>
      </review:hasReview>
      <dc:creator>
        <mm:Artist rdf:about="http://musicbrainz.org/artist/
           e795e03d-b5d5-4a5f-834d-162cfb308a2c"/>
      </dc:creator>
    </mm:Album>
  </foaf:interest>
 </foaf:Person>
</rdf:RDF>
```

Listing 3.6: Example of a user's FOAF profile

Once the user profile has been created, and the system is able to adapt it to the new user's interests, the next step is to exploit user preferences to recommend interesting items. This is covered in the next section.

3.2.2 User profile exploitation

The next three sections are devoted to explain the tehcniques related with user profile exploitation, already introduced in the previous chapter. Now, we focus on how these methods have been applied for music recommendation. The remaining section introduces the motivation of playlist generation, and the different approaches that exist.

Collaborative filtering

Collaborative filtering techniques have been largely applied in the music domain. In 1994, Shardanand described the first music recommender based on this technique, named Ringo [Sha94]. It used the Pearson normalized correlation between user profiles to make recommendations. Since then, a lot of music recommenders have appeared.

Racofi (Rule Applying COllaborative FIltering) Music [ABB+03] combines collaborative filtering (based on users' ratings), and a set of logic rules based on Horn clauses. The rules are applied after users' ratings have been gathered. The five rating dimensions they define are: impression, lyrics, music, originality, and production. The objective of the rules is to prune the output of the collaborative filtering and promote the items that the user will be most familiar with. [ABB+03] examplifies a rule:

> "*If a user rates 9 the originality of an album by artist X then the predicted originality rating, for this user, of all other albums by artist X is increased by a value of 0.5*".

These kind of rules implicitly modify the ratings that a user has done previously. The *Indiscover* music recommender system[5] implements this approach.

Since the growth of available data in Internet, there have been several attempts that apply item–to–item (or model–based) collaborative filtering based on co–occurrence anaylsis of text found on the web[6]. In [CF00], Cohen and Fan shows how collaborative filtering based on data crawled from the net sometimes outperform traditional methods. The idea is to augment user data for collaborative filtering with data collected from web crawlers.

[5] http://www.indiscover.net

[6] We could have classified these methods as Context–based filtering too.

In this sense, during the last years, artist similarity based on co-occurrence on the web (e.g both artists appear to the same HTML page), or co-ocurrences of songs in playlists, have been studied in the MIR field. Computing these similarities is based on a matrix M where both rows and columns correspond to music titles (or artists) and $M(i, j)$ denotes number of times i and j appeared together (in the same compilation, in the same playlist, or in the same web page). Pachet et al. [PWL01] computed artist and title co-occurrences from radio sources and from a big database of CD compilations[7]. Anyway, the first attempts based on data crawled from the web were made by Whitman et al. [WL02] In this approach, user collections from the music sharing service *OpenNap* were analyzed. The similarity measure was, then, based on community metadata. In [ZF04], Zadel and Fujinaga investigated artist similarity given an artist seed. Results were obtained by using Google and Amazon APIs. Artist co-occurrences on web pages was introduced in [SKW05a]. The approach was based on creating special Google queries for every pair of artists. Conditional probabilities were taken into account with page count results. The output of the co-occurrence matrix was used to evaluate genre classification. Though, the artist data set used was rather small. The drawbacks regarding polysemy of some artists' names (e.g. *Kiss*, *Bush*, *Porn*) were solved by the same authors in [SKW05b]. Last but not least, in [BH05], Baumann and Hummel evaluated cultural similarity among artists based on crawled data from the web. They applied text mining methods: Part-of-speech tagging, term weigthing (based on tf/idf), and filtering the content of the webpage, to focus only on the actual music related content.

Finally, issues about collecting ground truth to evaluate artist similarity are presented in [EWAS02], [BH05] and [Pac05].

Content–based filtering

Content–based methods over the music domain have been used to rank music titles based on audio similarity. Recently, Pandora's system[8] have had a big impact on the music industry. Pandora's approach is based on manual descriptions of the audio content. We can read from Pandora's web site[9]:

> *"(...) our team of thirty musician-analysts have been listening to*

[7] Extracted from *CDDB*. Now the free version is known as *FreeDB*

[8] http://www.pandora.com

[9] http://www.pandora.com/corporate/index.shtml

> *music, one song at a time, studying and collecting literally hundreds of musical details on every song. It takes 20-30 minutes per song to capture all of the little details that give each recording its magical sound —melody, harmony, instrumentation, rhythm, vocals, lyrics . . . and more— close to 400 attributes! We continue this work every day to keep up with the incredible flow of great new music coming from studios, stadiums and garages around the country (. . .)"*

On the other hand time–constraints only allow to add to the system about 7000 songs per month[10]. This presents serious scalability problems. To index huge audio repositories, automatic analysis is a must. Thus, researchers on audio and signal processing have focused on automatically extracting descriptors from the audio (section 4.1.3 presents the latest achievements of some *meaningful* descriptors).

The first works related with music similarity focused on low–level descriptors, such as the Mel Frequency Cepstral Coefficients (MFCC). These approaches aimed at deriving timbre similarity. Foote proposed a music indexing system based on MFCC histograms [Foo97]. Aucouturier and Pachet presented, in [AP02], a gaussian model based on MFCC. They could generate playlists based on timbre similarity and some global constraints of the output playlist. Similarly, Logan and Salomon [LS01] modelled timbre similarity using MFCC, but used a more complex matching algorithm than Aucouturier and Pachet. Though, none of these methods capture information about long–term structure (such as: melody, ryhthm, harmony, etc.). To cope with this limitation, Tzanetakis [Tza02] extracted a set of features representing spectrum, rhythm and harmony (chord structure). All the features are merged into a single vector, to determine similarity. Finally, the problem of how to gather ground–truth for music similarity evaluation is outlined in [BLEW03]. For a complete overview on audio similarity the reader is referred to [Pam06].

There are some relevant music systems that rely on audio content similarity. These systems are devoted to explore, to visualise, and to discover items in a music collection. *Musicsurfer*[11] automatically extracts descriptions related to instrumentation, rhythm and harmony. Together with complex similarity measures, the descriptions allow music collection navigation in a flexible and

[10]From the Podcast interview to Tom Conrad, CEO of Pandora: http://www.twit.tv/itn6

[11]http://musicsurfer.iua.upf.edu/

efficient way, and it can also generate smart playlists based on global constraints (such as: tempo, tonality, etc.). *Playola*[12] is a music similarity browsing tool. It allows to find new music by moving around in a music "space". A user can give relevant feedback to the sytem. Furthermore, along all the available frameworks and tools for audio and music computing[13], *MA Toolbox* implements similarity measures for audio data [Pam04].

Regarding the music recommendation problem, once the audio has been described, and similarity among items can be computed (see section 2.4.2 for an overview of possible distance metrics) it is rather straight forward to recommend music titles or artists. Finally, relevance feedback for content–based music systems is presented in [HMI03].

Context–based filtering

The context–based method has not been largely explored in the music field. The most relevant work has been done by Baumann et al. (see [BH03], and [BH05]). The authors presented a multi–facted approach for music recommendation, that took into account content–based features extracted from the audio, the lyrics of the title and cultural information. Regarding the latter, using tf/idf over the crawled descriptions of artists were able to characterize them. The description of artists are the terms with highest normalized tf/idf value. This set of terms includes the most relevant nouns, adjectives and simple phrases, as well as un–tagged unigrams and bigrams.

Once the description of the items (i.e artists) is available, the usual cosine–based similarity among a user's profile and items can be used to predict artists recommendations. In this case user's profiles could be a set of terms (adjectives, nouns or phrases) that describes the interests of the user. As the authors proposed, this method should be used in combination with other metrics, such as: audio similarity, lyrics similarity, etc. Thus leading to a hybrid music recommender.

[12] http://www.playola.org/

[13] For a complete list of MIR related tools see http://www.music-ir.org/evaluation/tools.html

A paradigmatic use case: playlist generation

Playlist generation is one of the most important applications of music recommendation. All the methods presented before allow to create playlists based on user preferences. There are several ways to create playlists: shuffle (i.e random), based on a given song (or artist) seed, based on the analysis of song co-occurrence, based on user's music neighbours, or based on music similarity. And, there is the option to mix several of these approaches.

Interestingly enough, some experiments have been carried out to investigate serendipity in random playlists. Nowadays, shuffle is still the usual way to generate playlists on personal computers and portable music players. In [LVH05], Leong et al. study the serendipity property through shuffle playlists, and report some user experiences. They argue that shuffle can invest new meaning to a particular song. It provides opportunities for unexpected rediscoveries, and re–connects songs with old memories. Although, we believe that serendipity can be achieved, too, by creating more clever playlists.

The assumption that song co-occurrence in a playlist means that these songs are *similar* is arguable. What if the playlist was created randomly? Moreover, the co-occurrence matrix —where both rows and columns correspond to music titles (or artists), and $M(i, j)$ denotes number of times i and j appeared together— is not normalised by the overall popularity of the item[14].

On the other hand, one could argue that playlists based only on content–based similarity have a very low coverage of the data collection. Thus, it could lead to a previsible and *boring* generation of playlists, without serendipity. We envision that merging different approaches (i.e. a hybrid recommender) when creating playlists, can improve user satisfaction.

There is a patent proposal from Apple about how to create personalized and smart playlists. This patent aims at selecting and playing music depending on the kind of exercise a person is doing. It can play a faster music to make a person runs quicker. The *http://www.unwiredview.com* blog[15] explains this idea:

> *(...) the invention pertains to a computing device that is capable of controlling the speed of the music so as to affect the mood and*

[14]This is similar to the tf/idf weighting idea. In the co-occurrence matrix only tf is taken into acount, whereas *idf* value normalises against the popularity.

[15]http://www.unwiredview.com/2006/05/25/ipod-sport-coming-from-apple-soon

> *behavior of the user during an activity such as exercise. By way of example, the speed of the music can be controlled to match the pace of the activity (synching the speed of the music to the activity of the user) or alternatively it can be controlled to drive the pace of the activity (increasing or decreasing the speed of the music to encourage a greater or lower pace). One aspect of the invention relates to adjusting the tempo (or some other attribute) of the music being outputted from the computing device.".*

To conclude, we foresee the exploitation of content–based descriptions as a way to enhance the interaction with playlist generation. In this sense, the logical unit of playlists are songs. Thus, playlist generation techniques are, to date, not ready to create a list of songs based on intra–song properties. For instance, a playlist based on a concrete *passage* of a song, or based on "music quotations" among songs. Nonetheless, this depends on the quality of intra–song descriptions, and the accuracy of song segmentation. Moreover, in the same feel as Apple's patent, there is an interesting research area about how to link user's experiences, moods, emotions, and situations with the audio content (and, by extend, to playlist personalization). [WPVS05] is another example, similar to Apple's idea. They present a personalized music system for motivation in sport performance, that includes a portable player with in-ear headphones, heart rate sensor belt, and pedometer using acceleration sensors.

3.3 Related systems

This section aims at describing some of the existing music recommenders. The overview of the systems is based on the elements presented in this chapter. Hence, for each system we describe (or conjecture) user profile representation (maintenance and adaptation), as well as the method(s) to exploit users' profiles. Moreover, we add more facets related to the music world. For instance, the diversity of the music library, the (estimated) number of users, data access from external third parties, etc.

The next sections present some systems that are representative, and that use the approaches presented before. We overview four systems: *Last.fm* —based on collaborative filtering—, *MyStrands* —mostly based on playlist co–occurrence—,

Pandora —based on manual descriptions of the music—, and *MusicIP Mixer* —based on automatic content–based descriptions. After that, we briefly present some other systems which aim at generating playlists.

Last.fm: a collaborative–filtering approach

The first sytem we overview is **last.fm**[16]. *Last.fm* system is based on the *Audioscrobbler* project. *Audioscrobbler* aims at tracking user listening habits on a personal computer. It offers a plug–in that works for most of the existing media players (and for Linux, Mac, and Windows operative systems). *Audioscrobbler* started in 2002, and is a clear example of exploiting music related data based on the collaborative–filtering approach.

Last.fm system provides a lot of functionalities: a downloadable client application that streams audio (based on user's preferences), discover neighbours with similar musical taste, visualize charts and basic statistics of user's listening habits, write and read music related blogs, participate on forums, search artists and songs by tags (also known as folksonomies), etc.

The representation of the profile consists mainly on the user listening habits (triplets that contain a timestamp, artist name and song title), gathered from the *Audioscrobbler* plug–in. Also, the profile contains a list of *loved* tracks and *banned* tracks, that probably are used when generating personalized playlists. The initial profile is empty, and the maintenance and adaptation is based on both implicit (tracking user listening habits) and explicit (love this song, ban this song).

User exploitation is based on collaborative–filtering. It allows both user profile–item matching (for instance: to generate playlists, and to get a list of recommended artists), and user profile matching (to create neighbourhoods). Playlist generation is based on user's neighours with similar musical tastes (more advanced features such as personal radio station is available only for subscribers). Moreover, a user can create a custom playlist based on a set of tags. These tags have been previously attach (by the users' community) to different artists and songs.

Their music collection seems not to be biased, and the system can play whole songs (not only excerpts). On the other hand, there is a big and active users'

[16]http://www.last.fm

community supporting the project. It is estimated that there are more than 1 million users, and the system gets more than ten million entries per day with regard to tracking users' listening habits. Most of the users have a free account, but upgrading user's account the system offers more personalized functionalities.

Last.fm data is available in several formats through the *Audioscrobbler* web services API, or via web syndication (through RSS feeds). The available information is: user profile data (e.g. recent songs played, top tracks, top artists, top albums, top tags, neighbours, etc.), artist data (e.g. related artists, top tracks, top albums, top tags), and track data.

Systems similar to *last.fm* are: *Musicmobs*[17] and *Yahoo! LaunchCAST*[18].

MyStrands: collaborative–filtering based on playlist co–occurrence

MyStrands[19] (previously known as *MusicStrands*) is another company involved in music recommendation. They started as a spin-off from the *Higher Council for Scientific Research and the Esfera* (Universitat Autònoma de Barcelona), in February, 2005.

The main application they provide is the **MyStrands** client. The application scans user listening habits from *iTunes* and *Windows Media Player* only. Based on the current song that is being played, *MyStrands* offers a list of similar songs, a set of tags related to explore their music catalogue, and a link to their website providing more information with regard to the actual song —as well as links to buy the music.

A user profile contain the user listening history, a set of handmade playlists (or imported from *iTunes* or *Windows Media Player* libraries), and the *exile list*. The exile list contains all the songs, artists or albums that a user has discarted.

The system uses playlist co–occurrence to determine similarities among songs, and this is basically how they exploit user profiles. Music recommendations (user profile–item matching) is based, then, on playlist co–occurrence. On the other hand, user profile matching —to get a list of neighbours— is computed too, based on user profile similarity. Relevance feedback to update the profiles is explicit.

[17] http://www.musicmobs.com
[18] http://launch.yahoo.com/
[19] http://www.mystrands.com

Users can discard recommended songs, albums or artists and move it to the exile list. The reasons to move an item to the exile list can be: because the user already knows the item (thus, it can be considered as an obvious recommendation), or because the user do not like that item.

The music collection contains both popular mainstream music, and *indie* artists, that submit their music and editorial information via *MyStrands* website. They claim that the size of the music catalogue is about five million titles. Although, when generating playlists the user can only preview a 30 seconds excerpt of a song.

Finally, they offer a web–based API, named *OpenStrands*, to access information about artists, albums, tracks and available playlists.

Pandora: generating manual content–based descriptions

The next system to overview is **Pandora**[20]. *Pandora* appeared in mid 2005 as a materialization of the *Music Genome Project*. The *Music Genome Project*, created in January 2000, is an effort from a group of musicians to "capture the essence of music at the fundamental level" by using over 400 attributes to describe songs. *Pandora* system offers a simple web–based music player, implemented in Acrobat Flash.

User profile representation contains a list of radio stations (created from a seed song or artist). Moreover, a bookmarked list of favourite artists and songs can be manually added by the user. Yet, it seems that the system does not take into account user's listening habits history.

The maintenance of the profile is based on explicit feedback (e.g thumbs–up/down for a given song that appears in a radio station). Probably, implicit feedback to improve radio stations is applied too. That is, when a song is skipped there is an adaptation process to decide the next songs to play. Interestingly enough, after a few number of skipped songs, the system ends up playing a song from the seed artist. This is to avoid user's frustration: after some songs being skipped, the system presents a song closely related with the seed, because it is probable that the user will like it (thus, not skipping it). Although, there is a limit of skips per hour, due to license agreements with their content providers. The music collection is biased toward popular music. There is no classical music,

[20]http://www.pandora.com

nor so–called world music. And, there is a lack of integration with personal off–line music collection.

User profile exploitation is based on manual content–based descriptions. This allows to generate radio stations (i.e playlists) with tracks similar to —or related with— the given seed song or artist. Furthermore, the system has the ability to explain to the user the reasons that made a song appear in a playlist (e.g. *"we're playing this track because it features mixed acoustic and electric rock instrumentation, a subtle use of vocal harmony (...)"*). This is an important property, and it gives credibility to the recommendation process. Moreover, in the *Pandora Backstage*, one can see artist relationships. It is not clear wheteher this is based on content–based features (inferred from songs, and aggregated at the artist level), or based on some sort of co–ocurrence analysis (plus users' relevance feedback) that is applied to the generated playlists.

Pandora has about two million users (data from May, 2006), and they claim that more than 70 million of relevance feedback entries have been reported by the users (until May, 2006). Although, there is no active users' community. This is not a problem per se, because the system is based on (manual) content–based analysis —thus, it does not rely on community data to generate playlists.

Data access is available through web syndication. Each user has a set of RSS feeds that contain the radio stations created, and the bookmarked artists and songs.

MusicIP Mixer: a content–based recommender

MusicIP[21] company, previously known as *Predixis*, has a desktop application (for Linux, Mac and Windows) that produces music recommendations from the user's music library. The application, named **MusicIP Mixer**, is based on the automatic audio analysis of the user's music titles. This application is a bit different from the ones explained before. It analyses all the audio files of the user's collection, and it provides playlist generation (or mixes) based on audio similarity.

User profile representation equals to the whole list of songs that the user owns. This is somehow extreme because it is quite common that a user only listens to fraction of his music library. Moreover, the system can import data from the

[21] http://www.musicip.com

iTunes application, such as: most played songs, most played artists. Yet, it is not clear what is the use of this data into the *MusicIP Mixer* application. To cope with the limitation of only recommending music from the user's library, the application has a link to *MusicIP Discover* webpage[22]. The user can query a song, artist or album. Once he has selected a song, the system retrieves a list of similar songs (most of them gathered from *CDBaby*[23] collection) based on audio features.

Another interesting functionality is the creation of moods. A user can create a certain mood, and assign it a label (e.g *happy music*), based on the songs that appear on a playlist —normally, created manually by the user himself. After that, the application can *learn* a given mood, based on the examples (i.e songs) given by the user, and it can generate new playlists based on a particular mood. Finally, users can improve their profile by explicit feedback (i.e ratings) on songs.

Other systems

Indiscover system[24] is based on the RACOFI technology [ABB+03]. Item–to–item matching implementation follows the (weighted) slope one method [LM05]. This method uses a regression model to derive similarities among items. From these similarities, *Indiscover* generates playlists based on song similarity (from users' ratings), or even based on moods (e.g. party, romantic, depressing, etc.). User profile representation mainly consist on user's song ratings. Music collection is biased toward independant music.

Another interesting system is **Tapestry**[25]. *Tapestry* generates playlists based on a set of descriptors from *AllMusicGuide*[26]. These descriptors range from cultural attributes: styles (era-specific types of music), themes (different concepts, activities, or environments), tones (the way the song feels, or the descriptive essence of the song), descriptors based on the audio itself: instrumentation and song structures (tempo, key, and dynamics), and editorial descriptors, such as the production elements (specific recording styles and techniques). *Tapestry* does not hold any representation of users' profiles, though.

[22] http://cdideas.predixis.com
[23] http://www.cdbaby.com
[24] http://indiscover.net
[25] http://tapestry.allmusic.com
[26] http://www.allmusic.com

MusicLens[27] system offers a playlist generator, based on a eclectic set of attributes, such as: volume, tempo, instrumental/vocal, purpose of the music (dance, driving, sex, etc.), voice gender (male/female), user's age, mood (sad, normal, happy), and decades. Each atttribute has a slider. When modifying a value, the system proposes a playlist based on the whole criterion. The interface, implemented in Acrobat Flash, is pretty clear and intuitive.

3.4 Summary and Conclusions

This chapter has presented the music recommendation problem. User preferences to state music interests is an important issue. We have presented three different notations: UMIRL, MPEG-7 based, and FOAF. The former is one of the first attempts in this field. UMIRL language is not formal enough, it is more a proposal that contains some interesting ideas. User preferences in MPEG-7 is the first big and serious attempt to formalise user modelling, related with multimedia content. The main problem of this approach is that the MPEG-7 standard is too complex and verbose. It is not straight forward to generate user profiles following the notation proposed by the standard. The last proposal, FOAF profiles, is based on the Semantic Web initiative. It is the most flexible approach. As it is based on the Semantic Web premises, FOAF profiles can embed different ontologies, so it is extensible, and has richer semantics than the other two approaches.

As we have seen in the overview of music recommendation systems (in section 3.3), user exploitation in music recommendation mostly follows the classic approaches presented in chapter 2. These systems provide ways to recommend artist and songs (when available) to the end–user. Yet, none of the studied systems offer other functionalities related with music discovery, such as: recommending podcasts, link with songs available from MP3 blogs, notification of upcoming concerts (based on user's musical taste), etc. We think that these functionalities are an added–value to a music recommender system.

Finally, we would like to mention some issues about the music data collections used by the recommenders. Quite often, the collection is biased. We have seen, in the overview of existing music recommenders, that there is a lot of Western music (sometimes even focused on US and UK mainstream music) and, normally,

[27] http://www.musiclens.de

only a few *other kind of music*. Thus, one might ask if there is a long–tail effect[28] that clearly bias the possible output recommendations.

3.4.1 Link with next chapters

Due to the complexity of describing the music field, an in–depth representation of the music objects are presented in the following chapters, 4 and 5. This characterization is important, because it allows to better understanding the complexity of music recommendations. Then, once the description of the music assets is available, a music recommendation can take advantage of this music knowledge representation. Chapter 4 is devoted to describe single music assets, whereas chapter 5 presents some multimedia ontologies that add semantics to the music objects.

[28]From the *Long Tail* weblog: "The Long Tail is the realization that the sum of many small markets is worth as much, if not more, than a few large markets" –proposed by Jason Foster (http://longtail.typepad.com/)

Chapter 4

Describing Music Assets

As outlined in the previous chapter, to characterise and describe music objects is not an easy task. There are several elements involved in that description. In this chapter we present the music information plane (MIP). This plane comprises most of the related information when describing music assets. MIP is presented in section 4.1. After introducing the MIP, the facets involved in describing music assets are presented. These facets include editorial metadata (described in section 4.1.1), cultural metadata (section 4.1.2), and acoustic metadata (section 4.1.3).

4.1 The Music Information Plane

We describe the music information plane in two dimensions. One dimension takes into account the different media types that serve as input data. The other dimension is the level of abstraction in the information extraction process of this data (see figure 4.1).

The input media types (in the horizontal axis) include data coming from: audio (music recordings), text (lyrics, editorial text, press releases, etc.) and image (video clips, CD covers, printed scores, etc.). On the other side, for each media type there are different levels of information extraction (in the vertical axis). The lowest level is located at the signal features. This level lays far away from what an end–user might find meaningful. Anyway, it is the basis that allow to describe the content and to produce more elaborated descriptions of the media objects. This level includes basic audio features (such as: energy, frequency, mel frequency cepstral coefficients, or even the predominant chord in a sequential

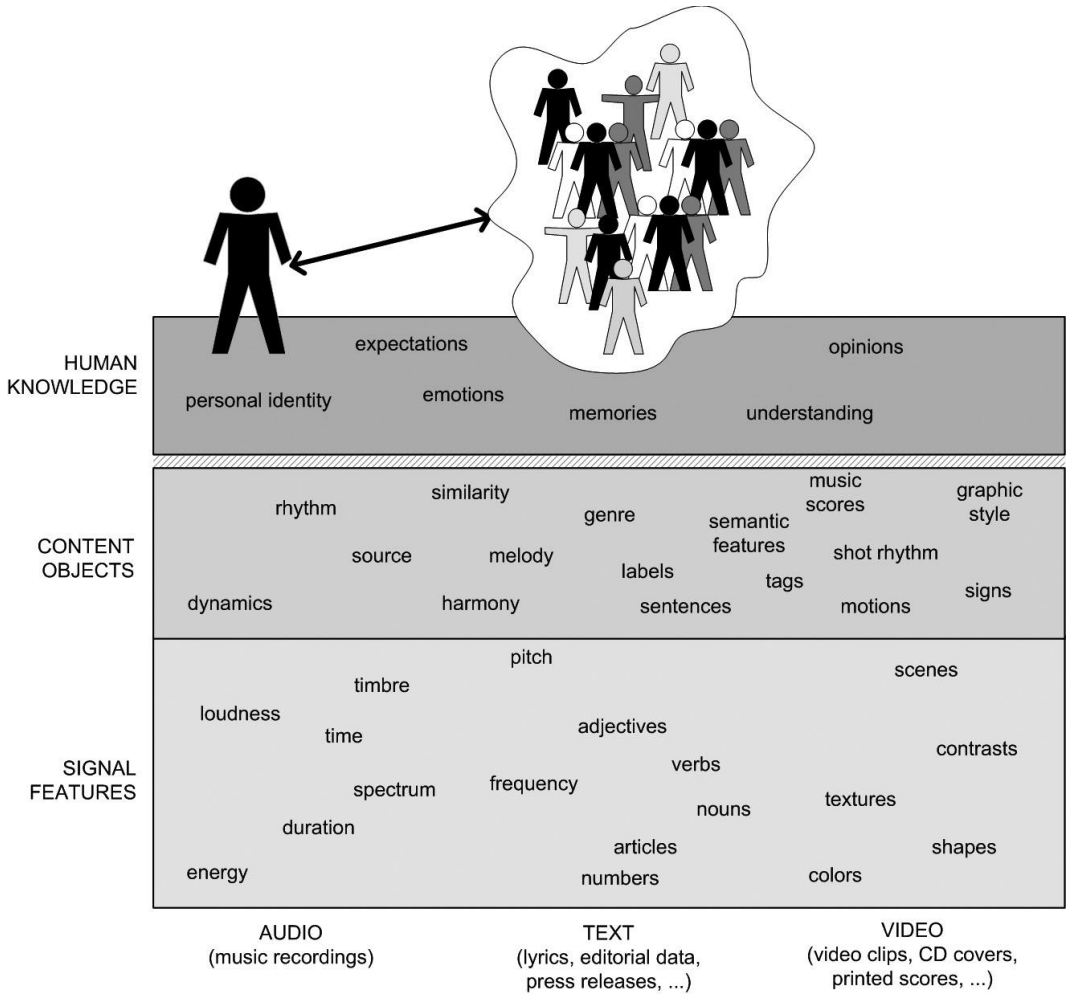

Figure 4.1: The music information plane. The horizontal axis includes the input media types. The vertical axis represents the different levels of information extraction for each media type. At the top, there is the user that interacts with the music content and with a social network of users.

list of frames), or basic natural language processing for the text media. At the mid–level (the content objects level), the information extraction process and the elements described are a bit closer to the end–user. This level includes description of musical concepts (such as a guitar solo, or tonality information —e.g key and mode— of a music title), or named entity recognition for text information. Finally, the higher–level, the human knowledge, includes information related with the human beings when interacting with music knowledge. This level could use inference methods and semantic rules to retrieve, for instance, several audio files with similar guitar solos over the same key. At the highest level, there is the user, and the social relationships with a community of users. Figure 4.1 depicts

the music information plane.

Nonetheless, the existing semantic gap between concept objects and human knowledge invalidates any possible direct assigment of music descriptors to users. This has a lot of consequences to music understanding (and music recommendation). Yet, there are some open questions, such as: what are the music elements that makes a person feel certain emotions, or to evoke some particular memories? How is personal identity linked with music? It is clear that only a multi–modal approach, that takes into account as much elements from MIP as possible, would be able to (partly) answer some of these questions. Furthermore, we argue that user intervention is crucial to add semantics to music understanding. That said, we believe that neither pure bottom–up nor top–down approaches can lead to bridge this gap. We foresee, then, an *approximation* in both ways: users need to interact with the content to add proper (informal) semantics, as well as content object descriptions must be somehow *understandable* by the users (that do not need to be experts on that domain). The next chapter arises some insights to this problem.

In [Pac05], Pachet classifies the music knowledge management. This classification allows to create meaningful descriptions of music, and to exploit these descriptions to build music related systems. The three categories that Pachet defines are: editorial, cultural and acoustic metadata. We include this classification as an orthogonal axis that lays over the MIP.

4.1.1 Editorial metadata

Editorial metadata (EM) consists of information manually entered by the editor. Usually, the information is decided by an expert or a group of experts. Figure 4.2 depicts the relationship between editorial metadata and the music information plane.

EM includes simple creation and production information (e.g. the song *C'mon Billy*, written by P.J. Harvey in 1995, was produced by John Parish and Flood, and the song appears as the track number 4, on the album "To bring you my love"). EM includes, in addition, artist biography, album reviews, genre information, relationships among artists, etc. As it can be seen, editorial information is not necessarily objective. It is usual the case that different experts cannot agree in assigning a concrete genre to a song or to an artist. Even more difficult

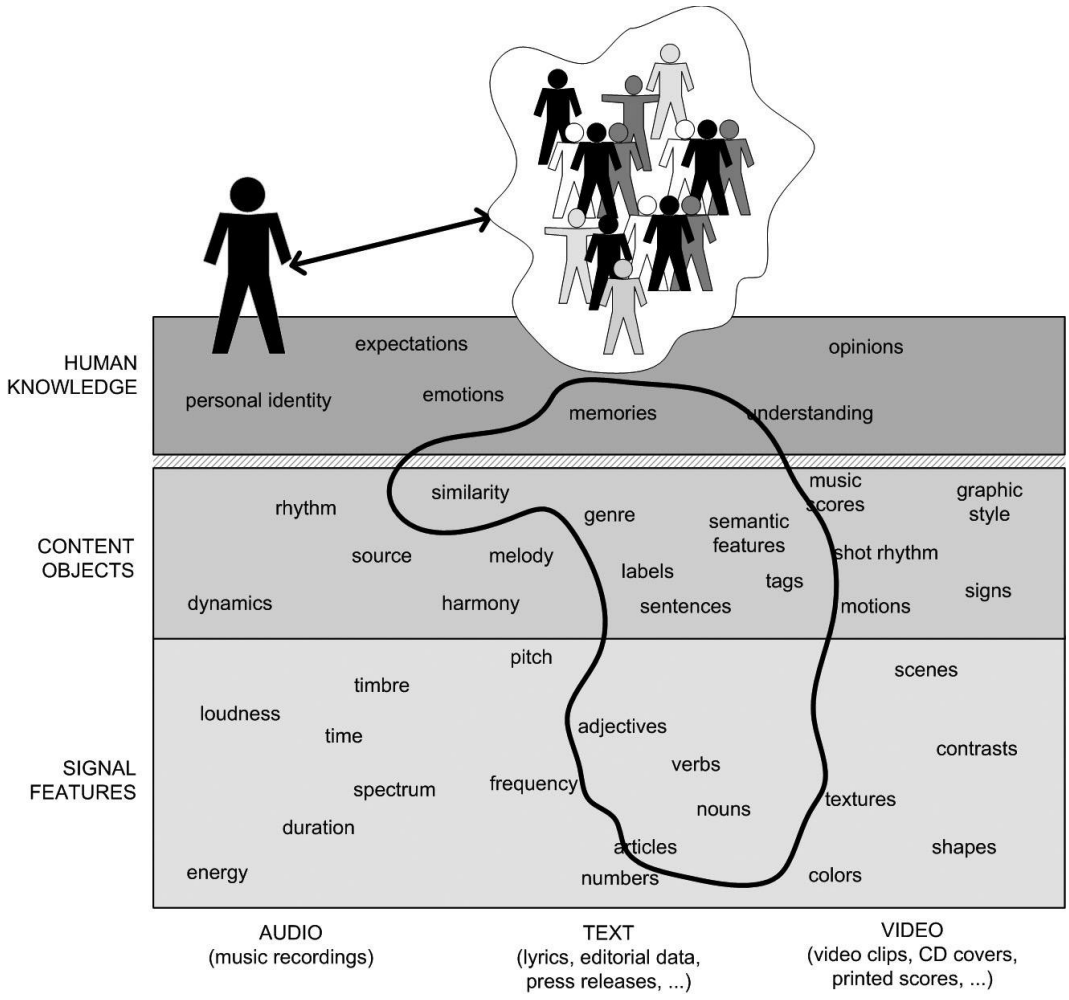

Figure 4.2: Editorial metadata and the music information plane.

is a common consensus of a taxonomy of musical genres.

EM conforms the core for collaborative–filtering music recommenders. As explained in section 3.2.2, collaborative filtering methods exploit this editorial information, and find relationships among them.

The scope of EM is rather broad. Yet, it usually refers to these items: the creator (or author) of the content, the content itself, and the structure of the content. Regarding the latter, editorial metadata can be fairly complex. For example, an opera performance description has to include the structure of the opera. It is divided in several acts. Each act has a number of scenes. In a given scene, there is a soprano singing an Aria piece, and a number of musicians playing. The Aria has the lyrics to sing. Lyrics can be in different languages (sung in Italian, but displayed in English), etc. The management of more complex

audiovisual descriptions is presented in chapter 5.

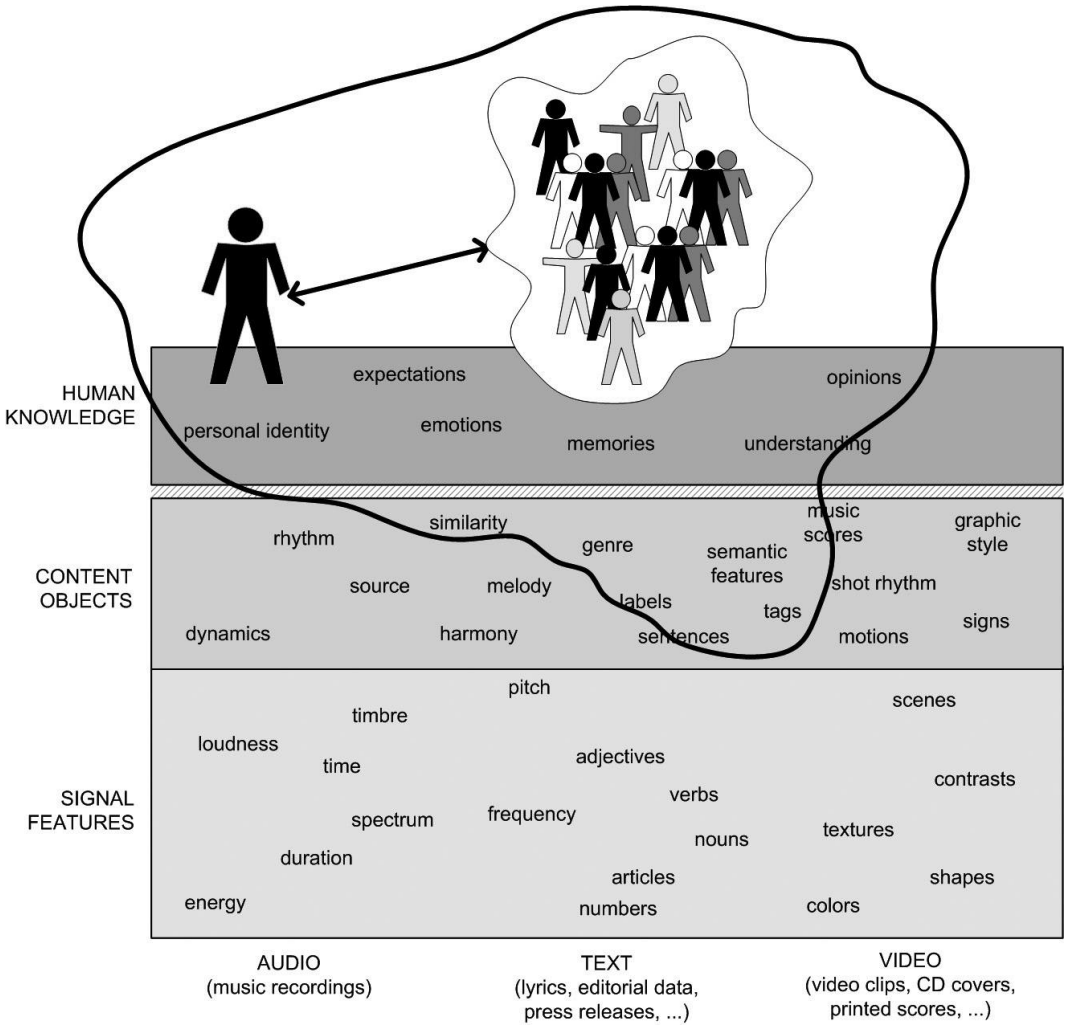

Figure 4.3: Cultural metadata and the music information plane.

4.1.2 Cultural metadata

Cultural metadata is defined as the information that is implicitly present in huge amounts of data. This data is gathered from weblogs, forums, music radio programs, or even from web search engines' results. This information has a clear subjective component as it is based on personal opinions. Figure 4.3 depicts the relationship between cultural metadata and the music information plane.

It is worth noting that this kind of information is the basis for context–based music recommenders and collaborative filtering (based on co–occurrences gathered from the web).

4.1.3 Acoustic metadata

The last category of music information is acoustic metadata. In this context, acoustic metadata describes the content analysis of an audio file. It is intended to be objective information. Obviously, acoustic descriptors are the basis for content–based music recommenders. Figure 4.4 depicts the relationship between acoustic metadata and the music information plane.

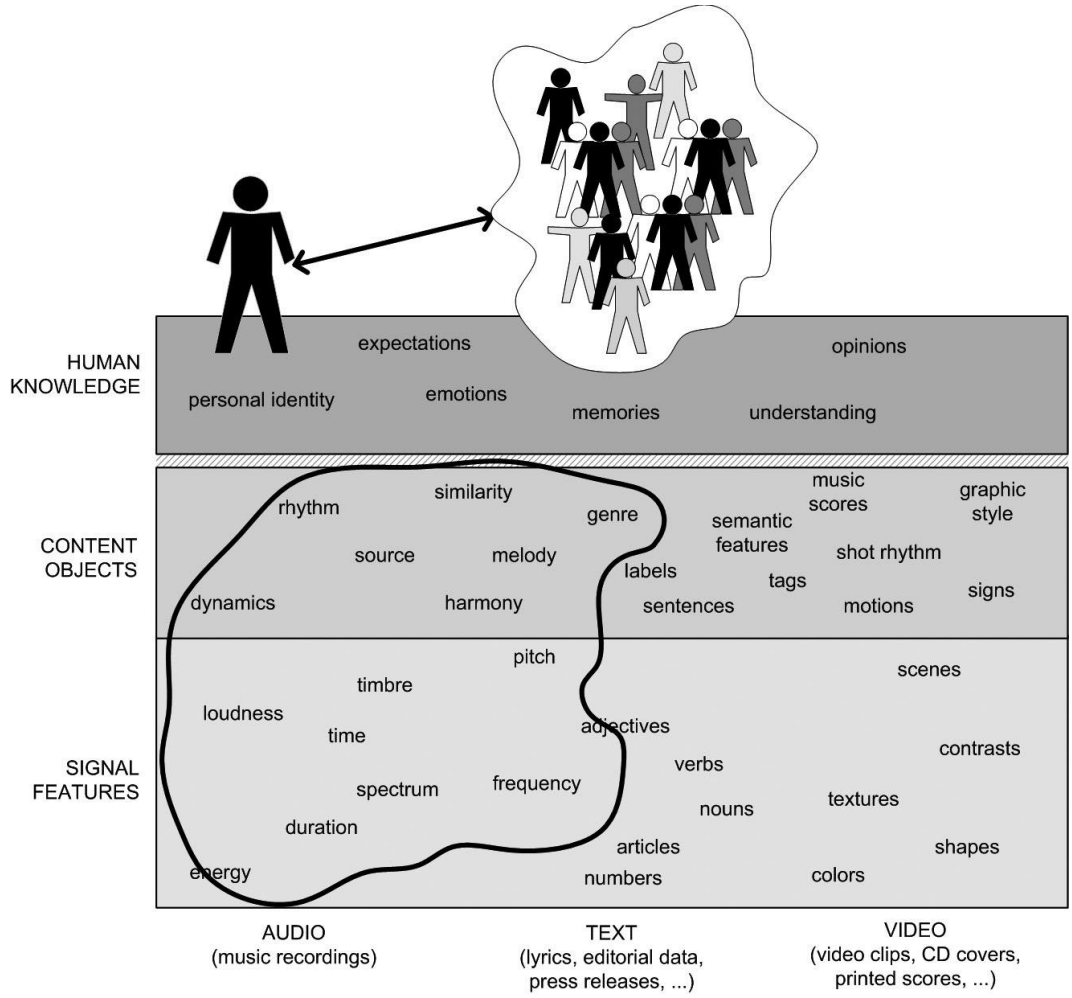

Figure 4.4: Acoustic metadata and the music information plane.

Most of the current music content processing systems operating on complex audio signals are mainly based on computing low–level signal features. These features are good at characterising the acoustic properties of the signal, returning a description that can be associated to texture, or at best, to the rhythmical attributes of the signal [AP02].

Alternatively, a more general approach proposes that music content can be

successfully characterized according to several "musical facets" (i.e. rhythm, harmony, melody, timbre) by incorporating higher–level semantic descriptors to a given feature set. Semantic descriptors are predicates that can be computed directly from the audio signal, by means of the combination of signal processing, machine learning techniques, and musical knowledge. Their goal is to emphasise the musical attributes of audio signals (e.g. chords, rhythm, instrumentation), attaining higher levels of semantic complexity than low–level features (e.g. spectral coefficients, Mel frequency cepstral coefficients, and so on), but without being bounded by the constraints imposed by the rules of music notation [Her06]. Describing musical content according to this view does not necessarily call for perfect transcriptions of music, which are outside the scope of existing technologies, even though recent outstanding progress has been reported in [HKD06].

Several of the shortcomings of the purely data driven techniques can be overcome by applying musical knowledge, and this musical knowledge should not be something exclusive for musically trained people. The richness of the description that can be achieved is well beyond that from existing music downloading and retrieval prototypes.

Current description schemes can be seen as a function of musical dimensions: rhythm, harmony, timbre and instrumentation, long–term structure, intensity, and complexity. The following sections are devoted to outlining recent achievements in music description. This multi–faceted description covers some open questions with regard to content–based analysis.

Rhythm

In its most generic sense, rhythm refers to all of the temporal aspects of a musical work, whether represented in a score, measured from a performance, or existing only in the perception of the listener [GD05]. In the literature the concept of "automatic rhythm description" groups a number of applications as diverse as tempo induction, beat tracking, rhythm quantisation, meter induction and characterisation of timing deviations, to name a few. A number of these different aspects have been investigated, from the low–level of onset detection, to the characterization of music according to rhythmic patterns.

At the core of automatic rhythmic analysis lies the issue of identifying the start, or onset time, of events in the musical data. As an alternative to stan-

dard energy–based approaches, another methodologies have recently appeared: a method that works solely with phase information [BS03], or that are based on predicting the phase and energy of signal components in the complex domain [BDDS04], greatly improving results for both percussive and tonal onsets. However, there is more to rhythm than the absolute timings of successive musical events. For instance, [DP04] have proposed a general model to beat tracking, based on the use of comb–filtering techniques on a continuous representation of "onset emphasis", i.e. an onset detection function. Subsequently, the method was expanded to combine this general model with a context–dependent model by including a state space switching model. This improvement has been shown to significantly improve upon previous results, in particular with respect to maintaining a consistent metrical level and preventing phase switching between off–beats and on–beats.

Furthermore, the work done by Gouyon ([GD04]) and Dixon ([DGW04]) demonstrates the use of high–level rhythmic descriptors for genre classification of recorded audio. An example is a tempo–based classification (see [GD04]) showing the high relevance of this feature while trying to characterize dance music. However, this approach is limited by the assumption that, given a musical genre, the tempo of any instance is among a very limited set of possible tempi. To address this, in [DGW04], an approach is proposed that uses bar–length rhythmic patterns for the classification of dance music. The method dynamically estimates the characteristic rhythmic pattern on a given musical piece, by a combination of beat tracking, meter annotation and a k–means classifier. Genre classification results are greatly improved by using these high–level descriptors, showing the relevance of musically–meaningful representations for Music Information Retrieval (MIR) tasks. Finally, a holistic approach toward automated beat tracking, taking into account music structure is presented in [Dan05].

For a more complete overview of the state of the art on rhythmic description towards a unified framework see [GD05].

Harmony

The harmony of a piece of music can be defined by the combination of simultaneous notes, or chords; the arrangement of these chords along time, in progressions; and their distribution, which is closely related to the key or tonality of the piece. Chords, their progressions, and the key are relevant aspects of music perception

that can be used to accurately describe and classify music content.

Harmonic based retrieval has not been extensively explored before. A successful approach at identifying harmonic similarities between audio and symbolic data was presented in [PBM+02]. It relied on automatic transcription, a process that is partially effective within a highly constrained subset of musical recordings (e.g. mono–timbral, no drums or vocals, small polyphonies). To avoid such constraints [Gom06b] adopts the approach where describes the harmony of the piece, without attempting to estimate the pitch of notes in the mixture. Avoiding the transcription step allows to operate on a wide variety of music. This approach requires the use of a feature set that is able to emphasise the harmonic content of the piece, such that this representation can be exploited for further, higher–level, analysis. The feature set of choice is known as a Chroma or Pitch Class Profile, and they represent the relative intensity of each of the twelve semitones of the equal–tempered scale.

In [GH04], Gómez presents the tonality estimation by correlating chroma distributions with key profiles derived from music cognition studies. Results show high recognition rates for a database of recorded classical music. The studies done in [HS05] have also concentrated on the issue of chord estimation based on the principled processing of chroma features, by means of tuning, and a simple template–based model of chords. Recognition rates of over 66% were found for a database of recorded classical music, though the algorithm is being used also with other musical genres. A recent development includes the generation of a harmonic representation by means of a Hidden Markov Model, initialized and trained using musical theoretical and cognitive considerations [BP05]. This methodology has already shown great promise for both chord recognition and structural segmentation.

For a complete and deeper overview of all these techniques, the reader is referred to [Gom06a].

Timbre and instrumentation

Another dimension of musical description is that defined by the timbre or instrumentation of a song. Extracting truly instrumental information from music, as pertaining to separate instruments or types of instrumentation implies classifying, characterizing and describing information which is buried behind many

layers of highly correlated data. Given that the current technologies do not allow a sufficiently reliable separation, work has concentrated on the characterization of the "overall" timbre or "texture" of a piece of music as a function of low–level signal features. This approach implied describing mostly the acoustical features of a given recording and gaining little abstraction about its instrumental contents [AP04].

Even though it is not possible to separate the different contributions and "lines" of the instruments, there are some interesting simplifications that can provide useful descriptors. Examples are: lead instrument recognition, solo detection, or instrument profiling based on detection without performing any isolation or separation . The recognition of idiosyncratic instruments, such as percussive ones, is another valuable simplification. Given that the presence, amount and type of percussion instruments are very distinctive features of some music genres and, hence, can be exploited to provide other natural partitions to large music collections. In [HSG04], Herrera et al. have defined semantic descriptors such as the percussion index or the percussion profile. Although they can be computed after some source separation, reasonable approximations can be achieved using simpler sound classification approaches that do not attempt separation [YGO04].

Additionally, [CDS05] contributed to the current state of the art in instrument identification of mono–instrumental music, using line spectral frequencies (LSF) and a k–means classifier. [HKD06]

Intensity

Subjective intensity, or the sensation of energeticness we get from music, is a concept commonly and easily used to describe music content. Although intensity has a clear subjective facet, Sandvold et al. hypothesized that it could be grounded on automatically extracted audio descriptors. Inspired by the findings of Zils and Pachet [ZP03], [SH04] created a model of subjective intensity built from energy and timbre low–level descriptors extracted from the audio data. They have proposed a model that decides among 5 labels (ethereal, soft, moderate, energetic, and wild), with an estimated effectiveness of nearly 80%. The model has been developed and tested using several thousands of subjective judgements.

Structure

Music structure refers to the ways music materials are presented, repeated, varied or confronted along a piece of music. Strategies for doing that are artist, genre and style–specific (i.e. the *A–B* themes exposition, development and recapitulation of a sonata form, or the *intro–verse–chorus–verse–chorus–outro* of "pop music"). Detecting the different structural sections, the most repetitive segments, or even the least repeated segments, provide powerful ways of interacting with audio content by means of summaries, fast–listening and musical gist–conveying devices, and on–the–fly identification of songs.

The section segmenter developed by Ong (see [OH05]) extracts segments that roughly correspond to the usual sections of a pop song or, in general, to sections that are different (in terms of timbre and tonal structure) from the adjacent ones. The algorithm first performs a rough segmentation with the help of change detectors, morphological filters adapted from image analysis, and similarity measurements using low–level descriptors. It then refines the segment boundaries using a different set of low–level descriptors. Complementing this type of segmentation, the most repetitive musical pattern in a music file can also be determined by looking at self–similarity matrices in combination with a rich set of descriptors including timbre and tonality (i.e. harmony) information [OH05]. Ground–truth databases for evaluating this task are still under construction, but first evaluations yielded an effectiveness of section boundary detection higher than 70%.

A final example

Finally, to recapitulate, the next example shows the description of an automatically annotated audio file, based on some of the descriptors presented in this section. There are descriptors that have an enumerated value as output —usually a label—, whereas other descriptors' values are numeric (e.g. floats or integers).

```
<?xml version='1.0' encoding='UTF-8'?>
<DescriptorsPool>
  <ScopePool name='Song' size='1'>
    <!-- EDITORIAL METADATA -->
    <AttributePool name='Title'>Last Salutation</AttributePool>
    <AttributePool name='Artist'>Randy Coleman</AttributePool>
    <AttributePool name='Duration'>247</AttributePool>
    <AttributePool name='Genre'>Pop</AttributePool>
    (...)
    <!-- ACOUSTIC METADATA -->
    <!-- Rhythm descriptors -->
```

```
    <AttributePool name='Tempo'>72</AttributePool>
    <AttributePool name='Measure'>
      <Enumerated>Binary</Enumerated>
    </AttributePool>
    <!-- Tonality descriptors -->
    <AttributePool name='Key'>
      <Enumerated>D</Enumerated>
    </AttributePool>
    <AttributePool name='Mode'>
      <Enumerated>Major</Enumerated>
    </AttributePool>
    <AttributePool name='Key-Strength'>0.8412</AttributePool>
    <!-- Intensity descriptor -->
    <AttributePool name='Intensity'>
      <Enumerated>Soft</Enumerated>
    </AttributePool>
    <AttributePool name='Danceability'>
      <Enumerated>Few</Enumerated>
    </AttributePool>
    (...)
  </ScopePool>
</DescriptorsPool>
```

Listing 4.1: Example of an automatically annotated audio file.

Example 4.1 shows a description of a music title. Based on this description a music recommender is able to, for instance, generate a playlist based on acoustic and editorial criterion. One of our approaches to music recommendation makes use of these descriptions, plus cultural metadata, and low–level acoustic information. This prototype, named *Foafing the Music*, is presented in chapter 7.

4.2 Summary and Conclusions

In this chapter we have introduced different approaches to describe music facets, based on the music information plane (MIP). As it can be seen in figure 4.5, editorial, cultural and acoustic knowledge conforms a fuzzy classification. There are some parts of the MIP that overlap. Of special interests are *similarity* and *genre* concepts. All three metadata descriptions tackle these two concepts. This fact links with the music recommendation problem, and the approaches presented in section 3.2.2 (collaborative filtering, content–based and context–based filtering).

Moreover, there is one part of the MIP that is not covered by the classification, that is the one related with video and images. This niche could represent some

loss of information to describe music objects. Yet it is not clear how this media type could link with text and audio, and enhance music recommendations, but still, we foresee that is important to be taken into account[1]. For instance, how can user's images (available from his profile) be exploited to recommend music?

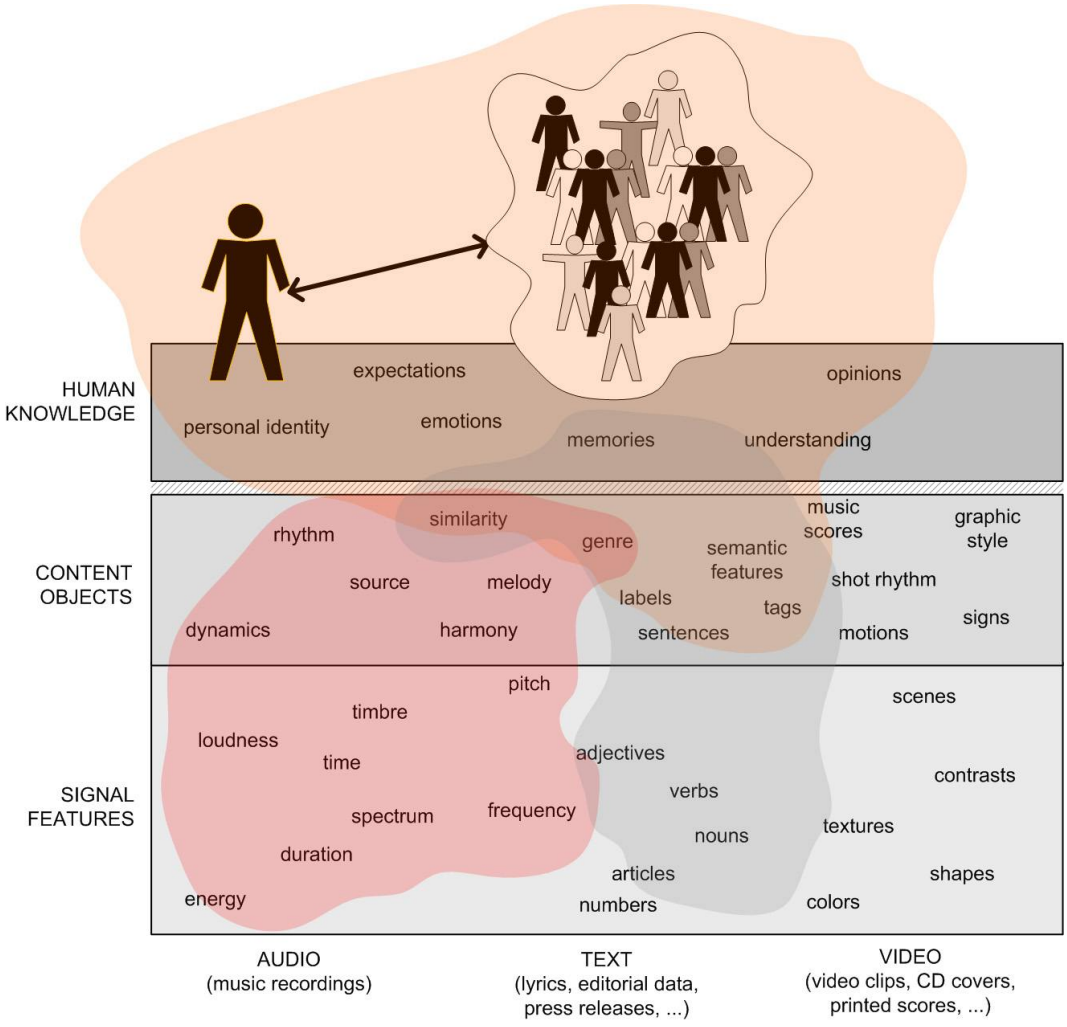

Figure 4.5: Music knowledge management (editorial, cultural and acoustic metadata) lay over the music information plane. *Similarity* and *genre* concepts are tackled by the three metadata descriptions.

[1]Remember the story at the introduction (section 1.2), where the protagonist discovers his all–time favourite band via an album cover.

4.2.1 Links with music recommendation

Describing music assets is a crucial task for any music recommender system. The success of a music recommender can depend on the accuracy and level of detail of this information. A music recommender that uses as much information as possible from the domain can attain better recommendations, because it will have a better knowledge representation. On the other hand, this can lead to a more complex algorithms to derive recommendations.

Chapter 5

Managing Audiovisual Descriptions

This chapter is devoted to the management of audiovisual content. More concretely, the chapter aims at adding explicit semantics to the music descriptions presented in the previous chapter via ontology management. The first section, 5.1, explains the motivation and importance of using formal ontologies for knowledge representation. It briefly presents Dublin Core, Wordnet, MPEG-7 and OWL. After that, and extensive overview of the MPEG-7 multimedia standard is done in section 5.2. Section 5.3 presents the ontology vocabularies used in the context of the Semantic Web. Then, section 5.4 links the MPEG-7 standard and the Semantic Web approaches, and presents the method that we have used to move MPEG-7 descriptions to the Semantic Web world. Based on this approach, some useful applications —such as: integration of metadata from different sources, and propagation of annotations— are outlined.

5.1 Motivation

During the last decades, digital media has been a revolution for media reproduction. This, in combination with the media distribution break–up carried out by the World Wide Web, has produced an explosion of the media availability. The amount of digital media that has been generated and stored, and which continues to do so at an exponential rate, has already become unmanageable without fine–grained computerised support. Low–level approaches, based on signal analysis, are proving to be extremely limiting in making multimedia database

systems accessible and useful to end–users. These content based descriptors lay far away from what users recognise as media description means. Consequently, recent research has begun to focus on bridging the semantic and conceptual gap that exists between user and computer —from content-based to high-level descriptions. The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation [SWS+00].

Searching in digital libraries has been widely studied for several years, mostly focusing on retrieving textual information using text–based methods. These queries can be complemented and improved with advanced retrieval methods using content based descriptors (extracted from the audiovisual information by applying signal processing —see section 4.1.3— and machine learning techniques). Even though some knowledge management and representation of the content is necessary. Moreover, from the service and content providers' point of view, multimedia metadata represents an added–value to audiovisual assets, but then again manual annotation is a labor–intensive and error–prone task. Thus, managing audiovisual essence implies to structure its associated metadata; using description schemes, taxonomies and ontologies, to organize a meaningful data knowledge representation.

Due to the inherent complexity to describe multimedia objects, a layered approach with different levels of granularity is needed when designing an ontology for a particular domain. Depending on the requirements, one might choose the appropriate level of abstraction. In this chapter we classify different audiovisual related ontologies based on its expressivity. Figure 5.1 depicts the expressiveness of several description languages relevant for representing (multimedia) metadata.

Dublin Core[1] is a controlled vocabulary to describe editorial information of documents. It contains 15 basic terms (title, creator, description, etc.). This vocabulary only allows to describe editorial information (see section 4.1.1). Actually, it was designed with the objective of describing editorial information of digital libraries. Next example shows the metadata description of a song:

```
<dc:title>C'mon␣Billy</dc:title>
<dc:creator>P.J.␣Harvey</dc:creator>
<dc:date>1995</dc:date>
<dc:language>en</dc:language>
<dc:type>sound</dc:type>
```

[1] http://dublincore.org/

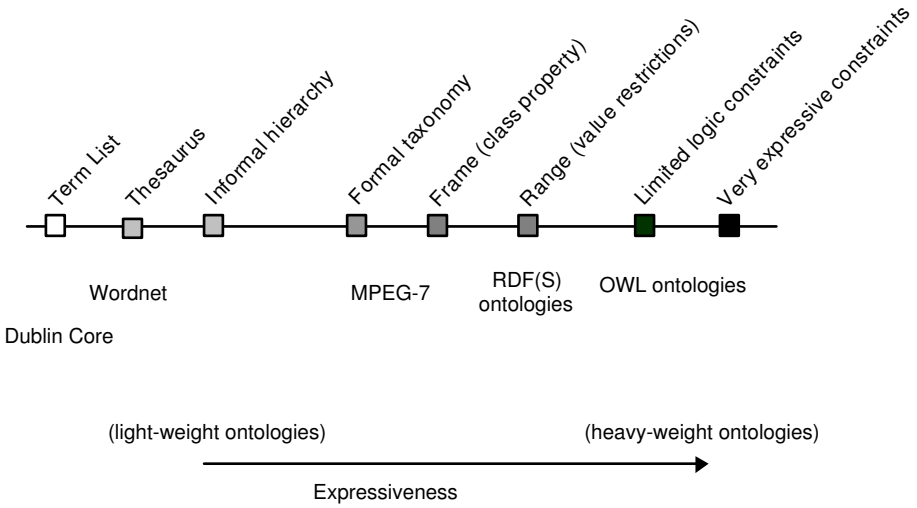

Figure 5.1: Classification of ontology languages based on its expressiveness (partially adapted from [dB03])

```
<dc:format>audio/mpeg<dc:format>
```

Listing 5.1: Metadata description of a song using Dublin Core.

WordNet[2] is a semantic lexicon (a thesaurus) that defines a set of relationships among words, such as: synonyms, antonyms, meronyms and hyponyms [Mil95]. In this context, [CKH+04] presents an all–purpose sound recognition system based on nearest–neighbor classification rule. A sound sample is labeled with the descriptions from the similar sounding examples of an annotated database. The terms borrowed from the closest match are unambiguous due to the use of WordNet as the taxonomy back–end. WordNet taxonomy allows, then, to describe editorial metadata and basic acoustic information (based on the propagation of annotations). For example, a violin sound with the following caption: *"violin pizzicato D#"* has the following synonym ring (or Wordnet synsets) [CKH+04]:

1. **violin**, **fiddle** – (bowed stringed instrument that is the highest member of the violin family; this instrument has four strings and a hollow body and an unfretted fingerboard and is played with a bow)

2. **pizzicato** – ((of instruments in the violin family) to be plucked with the finger)

[2]http://wordnet.princeton.edu/

3. **re**, **ray** – (the syllable naming the second (supertonic) note of any major scale in solmization)

4. **sharp** – ((music) raised in pitch by one chromatic semitone; "C sharp")

Another proposal is the MPEG-7 standard. MPEG-7 is largely presented in section 5.2, and the proposals in the context of the Semantic Web —the Resource Description Framework Schema, RDF(S), and the Web Ontology Language[3] (OWL)— are outlined in 5.3. All these proposals can describe editorial, cultural, and acoustic information.

5.2 Overview of the MPEG-7 standard

MPEG-7, formally named *Multimedia Content Description Interface*, is an ISO/IEC standard developed by the Moving Picture Experts Group (MPEG), the committee that also developed the audiovisual standards: MPEG-1, MPEG-2, MPEG-4 and MPEG-21. MPEG-7 aims to create a standard for the description of the multimedia content data. The main goal of the MPEG-7 standard is to provide structural and semantic description mechanisms for multimedia content [MS02].

MPEG-7 standard provides content description for audiovisual content, defining normative elements as *Descriptors*, *Description Schemes* and a *Description Definition Language* (DDL). DDL constitutes the basic building blocks for the MPEG-7 metadata language. Descriptors are designed for describing different types of information; low–level audiovisual features, high–level semantic objects, content management and information about storage media. Description Schemes are used to group several Descriptors (and Description Schemes) into structured semantically units using the DDL. Ideally, most descriptors corresponding to low–level would be extracted automatically, whereas human intervention would be required for producing high–level descriptors.

The standard is divided into four main components: the DDL, the Audio part, the Visual part, and the information about how these elements are combined in a multimedia scenario —a set of Multimedia Description Schemes that includes all the descriptors for capturing the semantic aspects of multimedia contents,

[3]http://www.w3.org/2004/OWL/

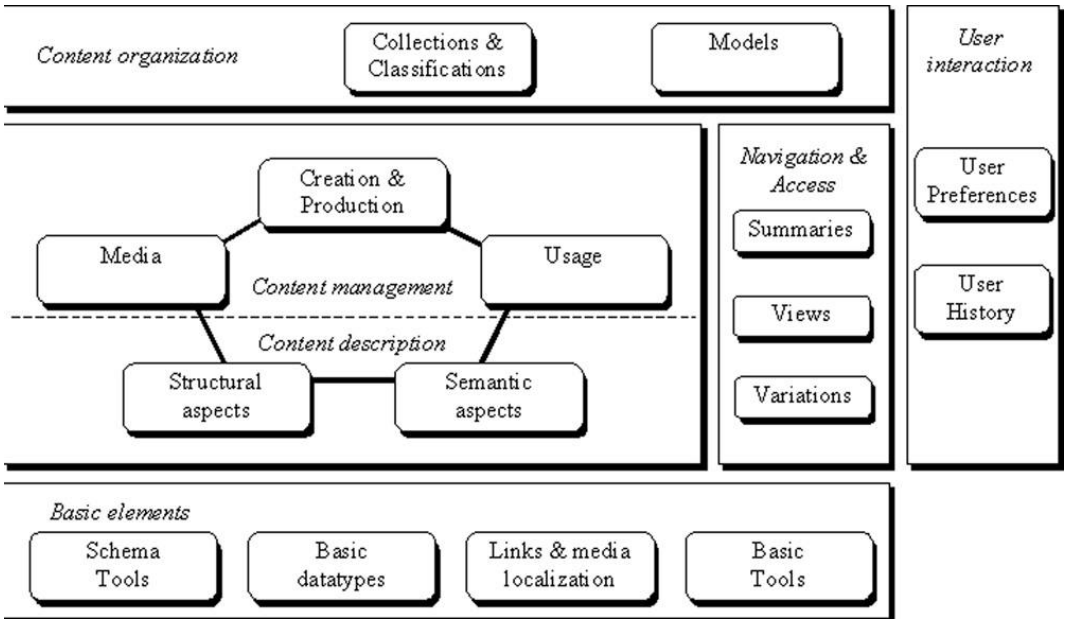

Figure 5.2: Main elements of the MPEG-7 Multimedia Description Schemes

e.g. places, actors, objects, events, etc. Thus, the creation of MPEG-7 documents allow a user to query and retrieve (parts of) multimedia and audiovisual information.

5.2.1 Multimedia Description Schemes

Of special interest is the part 5 of the standard, named *Multimedia Description Schemes* (MDS). This part includes a set of description tools dealing with generic features and multimedia descriptors. Figure 5.2 depicts all the components of the MDS. The Basic elements component includes basic datatypes, such as media localization, time format, free text annotations, etc. It includes, also, the classification schemes (CS) descriptors. CS descriptors define a scheme for classifying a subject area with a set of terms, organized into a hierarchy (i.e a taxonomy). Similar to the WordNet linguistic ontology, basic relationship among the taxonomy terms are available (e.g. narrow and broader terms, and synonyms).

One of the main components of MDS is the *Content Management and Description schemes*. Content Management descriptors allow to describe the life cycle of multimedia content, from its creation to its usage. It includes Media Information to describe storage format, media quality, media location, etc. Moreover, Content Managing schemes allow to gathering editorial data about

the creation and production process of the content. Content Description schemes describe the structural aspects (spatial, temporal and media source structure of multimedia content) and the semantic aspects.

The definition of the semantic description tools is one of the main caveats of the standard. As a mere syntax language, the DDL does not provide methods for subsumption reasoning on the class hierarchy [ONH03]. Section 5.4 presents a methodology to solve this limitation.

5.2.2 MPEG-7 and multimedia database systems

[Kos04] defines an architecture for a multimedia database management system that includes MPEG-7 descriptions, as well as the streaming service of the associated audiovisual files. An architecture of a multimedia database, inspired in [Kos04], is depicted in figure 5.3.

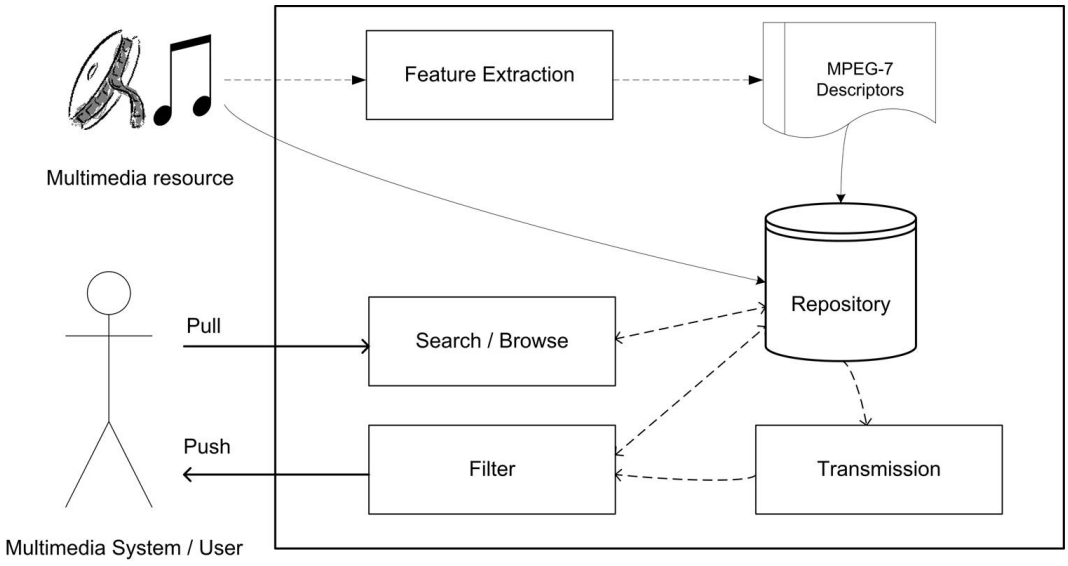

Figure 5.3: General architecture of a Multimedia Database Management System (based on [Kos04]).

Starting from the feature extraction and annotation process of a multimedia asset, the MPEG-7 descriptors are generated and stored in a repository. Typically, in a multimedia database system one can distinguish two query scenarios: *pull* and *push*. In a *pull* scenario, a user submits queries to the system and receives a set of descriptions satisfying the constraints of the query. On the other hand, in a *push* scenario, a software agent selects MPEG-7 descriptions and per-

forms a set of actions afterwards. One of these actions could be, for instance, proposing to users media information and its description based on their preferences. Hence the user agent is filtering audiovisual information according to metadata description.

There are several approaches and paradigms for structuring MPEG-7 information into a database system. We point out two general solutions: *(i)* to model MPEG-7 data into a relational database system and *(ii)* to use a native XML database (XML:DB)[4]. The former is based on the classic concept of a relation, while the latter has XML documents as its fundamental unit of (logical) storage in the database. The next two sections explain both approaches.

Relational Databases

The work that has been previously done for structuring MPEG-7 data into a database system is based, mostly, on the classical relational model —plus some extensions to adapt the XML information into the relations. For instance, Jacob [Jac04] has implemented a database to manage descriptions of sound objects —in MPEG-7— using a PostgreSQL database. A set of extensions (mainly programming triggers) has been designed to take into account insertions, updates and deletions of elements in an MPEG-7 document. An extraction rule engine allows to generate the MPEG-7 data. Yet, it is not clear, from a database user point of view, how to query (*select*) the MPEG-7 data inside the DB. Doller and Kosch [DK03] have designed an *MPEG-7 Multimedia Data Cartridge*. This system is an extension of object–relational DBMS Oracle 9i, providing a multimedia query language, access to media, and indexing capacities. Descriptors in the MPEG-7 schema are mapped to object types and tables, thus allowing to express queries in an hybrid *SQL* and *XPath* language.

Both systems permit to validate XML elements with the XML Schema (i.e validating MPEG-7 descriptions using the MPEG-7 DDL), providing a way to assure data integrity. However, due to the fact that the MPEG-7 DDL is tightly associated to the XML Schema definition, and the difficulties of reverse–engineering the model, managing MPEG-7 descriptors is equivalent to managing XML documents [WK03]. Thus, in this approach there is a big of effort in transducing the whole MPEG-7 Schema within a set of relations (i.e tables). Even so, possi-

[4]Its definition is available at: http://en.wikipedia.org/wiki/XML_database

ble changes on the MPEG-7 standard would imply to redo part of the database schema, which might be unfeasible when a system is being exploited. To cope this problem in a general sense, there has been considerable research concerning the automatic mapping between schema definitions of XML documents and relational database schema ([STH+99], [TDCZ02]), but most of the work is focused on DTD definitions instead of XML Schema[5], so they do not suffice for the management of MPEG-7 data.

Native XML Databases

Another approximation to structuring MPEG-7 descriptors is to use a native XML:DB. According to Bourret[6], there are more than 35 (open source and commercial) native XML database systems. Native XML:DB define a (logical) model for an XML document, and stores and retrieves documents according to that model. Native XML:DB make use of collections as internal folders for repositories of XML documents.

Westerman et al. have reviewed, in [WK03], the existing database systems that can manage MPEG-7 media descriptions. Their study includes a set of native XML database. A table comparision between systems unveils a lack of data integrity validation by most of them. Data integrity is a key point in any database system. In a native XML:DB integrity validation is done by parsing an XML document through its schema definition. Data integrity should be verified after an insertion of a new XML document to the DB, or after a modification of an already existent document. None of the native XML:DB presented in [WK03] allow full schema validation of MPEG-7 descriptors through MPEG-7 DDL. To solve this issue, a proposal of schema validation applied to native XML:DB is presented in [HL02].

As native XML:DB are still reaching maturity, another important aspect is to define languages that allow to query, insert, update and delete elements in the document. The most used languages —by XML:DB implementations— to query and to retrieve (part of) documents are the W3C *XPath 2.0*[7] and *XQuery 1.0*[8] recommendations. *XQuery* is a functional, strongly typed language that

[5]The critical difference between DTDs and XML Schema is that XML Schema uses an XML–based syntax, whereas DTDs have a unique syntax held over from SGML DTDs

[6]http://www.rpbourret.com/xml/XMLDatabaseProds.htm

[7]http://www.w3.org/TR/2004/WD-xpath20-20040723

[8]http://www.w3.org/XML/Query

satisfies the requirements of a database query language. Updating XML data is possible with *XUpdate* initiative[9]. *XUpdate* is a simple XML update language. It can be used to modify XML content by simply declaring, in an XML syntax, what changes should be made. Next section explains how to query MPEG-7 documents using the *XQuery* language.

Retrieving information from MPEG-7 descriptions

The eXtensive Markup Language[10] (XML) has been adopted as the format to represent MPEG-7 descriptors. Also, MPEG-7 DDL is an extension of the W3C XML Schema[11]. XML Schema provides the means for defining the structure of XML documents, that is; simple and complex data types, type derivation and inheritance, element ocurrence constraints and, finally, namespace–aware for elements and attributes declarations. MPEG-7 DDL extends the XML Schema and covers the ability to define array and matrix datatypes, and provides specific temporal descriptions (by means of the *basicTimePoint* and *basicDuration* types).

The set of MPEG-7 XML Schemas defines 1182 elements, 417 attributes and 377 complex types. The size of this standard makes it quite difficult to manage. Moreover, the use of XML technologies implies that a great part of the semantics remains implicit (see section 5.2.1). Therefore, each time an MPEG-7 application is developed, semantics must be extracted from the standard and re–implemented.

Next two examples depict how to retrieve information from MPEG-7 MDS documents using the *XQuery* language and an XML database[12]. First example 5.2 shows an expression to retrieve MPEG-7 audiovisual segments containing any media information. The ouput is presented as simple HTML code, containing a link to the media file —with the title and type of file as the text link.

```
for $segment in //AudioVisualSegment
let $title:= $segment/CreationInformation/Creation/Title/text()
order by $title
return
  for $media in $segment/MediaInformation/MediaProfile
```

[9] http://www.xmldatabases.org

[10] http://www.w3.org/XML/

[11] http://www.w3.org/XML/Schema

[12] These examples where tested using the eXist XML:DB available at http://www.exist-db.org/

```
  let $file:=$media/MediaInstance/MediaLocator/MediaUri/text()
  let $type:=$media/MediaFormat/Content/Name/text()
  return
    <a href="{$file}"> { $title , "␣[" , $type , "]" } </a>
```

Listing 5.2: *XQuery* expression to retrieving a list of multimedia items (title and format type).

The second example shows an *XQuery* expression to retrieve all MPEG-7 person agents, whose role is *Singer*, and the characters they play. This query uses a taxonomy that defines different type of singers' role (soprano, contralto, tenor and bass):

```
for $creator in
 /Mpeg7/Description/MultimediaContent/*/CreationInformation/
    Creation/Creator
where
  $creator/Role[@href="urn:opendrama:cs:SingerCS:%"]
 and
  $creator/Agent[@xsi:type="PersonType"]
order by $creator/Agent/Name/FamilyName
return
  <agent>
  {
   let $completeName:= $creator/Agent/Name
   let $name:= $completeName/GivenName/text()
   let $surname:= $completeName/FamilyName/text()
   return
    <singer> { $name , "␣" , $surname } </singer>
  }
  {
   let $completeName:= $creator/Character
   let $name:= $completeName/GivenName/text()
   let $surname:= $completeName/FamilyName/text()
   return
    <character> { $name , "␣" , $surname } </character>
  }
  </agent>
```

Listing 5.3: *XQuery* example to retrieving the singers and the characters they play.

Previous examples only illustrate one kind of difficulty derived from the use of just syntax–aware tools. In order to retrieve any kind of MPEG-7 *SegmentType* descriptions from an XML database, one must be aware of the hierarchy of segment types and implement an *XQuery* that covers any kind of multimedia segment (i.e. *AudioVisualType VideoSegmentType*, *AudioSegmentType*, etc.). On the other hand, once the hierarchy of segments is explicitly defined in an ontology (e.g in OWL form), semantic queries benefit from the, now, explicit

semantics. Therefore, a semantic query for *SegmentType* will retrieve all the subclasses without requiring additional efforts. This is necessary because, although XML Schemas capture some semantics of the domain they model, XML tools are based on syntax. The captured semantics remain implicit from the XML processing tools point of view. Therefore, when an *XQuery* searches for a *SegmentType*, the *XQuery* processor has no way to know that there are many other kinds of segment types that can appear in its place, i.e. they are more concrete kinds of segments. At this stage, a possible solution to avoid this is to use wildcards' syntax (see second and fifth lines of the example 2.2).

Anyway, MPEG-7 constitutes a valuable starting point for more specific developments as it can be seen as an "upper–ontology" for multimedia. However, the lack of explicit semantics makes MPEG-7 very difficult for third party entities to extend in an independent way. This lack of facilities for easy extension has been one of the main motivations to build solutions that make MPEG-7 semantics formal and thus easily machine–processable. Some solutions to this problem are detailed in section 5.4.

5.3 Web Ontology Languages

The World Wide Web has changed the way people communicate with each other. Most of todays' Web content is suitable for human consumption. Keyword–based engines have helped users to find the information they are seeking on the net. Yet, search engines present some limitations [AvH04]: the results are single web pages, results are highly sensitive to the vocabulary (semantically similar queries should return similar results), and usually there is a high recall and low precision of the result set (i.e there is too much noise on the webpage results).

The main problem of the current Web, at this stage, is that the meaning of the content is not accessible by machines. Information retrieval and text processing tools are widely used, but there is still difficulties when interpreting sentences, or extracting useful information for users. The development of the Semantic Web, with machine–readable content, has the potential to revolutionize the current World Wide Web and its use.

5.3.1 Overview of the Semantic Web

The definition and vision that had Tim Berners Lee, back in 1999 ([BL99]), is that the Semantic Web is an extension of the current web in which information is given well–defined meaning, better enabling computers and people to work in cooperation. The Semantic Web is a vision: the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications ([BLHL01], [SLH06]).

The previous ideas and principles to enhance the Web are being put into practice under the guidance of the World Wide Web Consortium (W3C). The next statement presents their view:

> *"The semantic web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The mix of content on the web has been shifting from exclusively human–oriented content to more and more data content. The semantic web brings to the web the idea of having data defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications. For the web to reach its full potential, it must evolve into a semantic web, providing a universally accessible platform that allows data to be shared and processed by automated tools as well as by people."* — W3C Semantic Web Activity Statement

The Semantic Web technologies have been arranged into a layered architecture. The key technologies include explicit metadata, ontologies, logic and inferencing, and intelligent agents. Each layer, from the bottom to the top, has an increasing level of complexity, yet it offers more expressivity. Figure 5.4 depicts the proposed architecture of the Semantic Web.

The two base layers (Unicode and URI, and the XML family) are inherited from the current Web. Section 5.2 already has presented some technologies related with XML. The upper layers compose the Semantic Web, over the existing basic technologies. The next sections explain the Resource Description Framework (RDF) and the RDF Schema, and the Ontology Vocabulary (OWL).

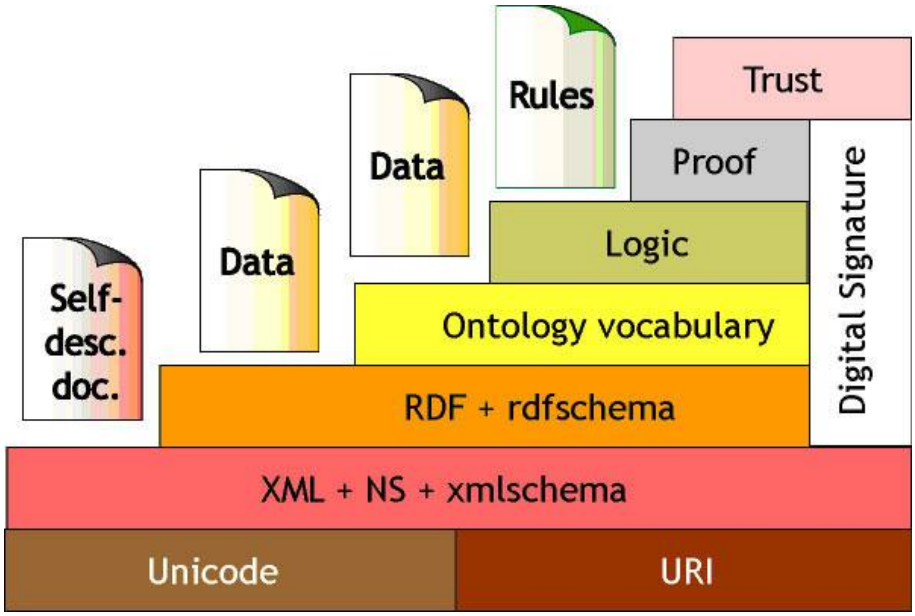

Figure 5.4: A layered approach to the Semantic Web.

5.3.2 Resource Description Framework

RDF vocabulary is based on the idea of conceptual graphs (CG) or semantic nets. CG express meaning in a form that is logically precise, humanly readable, and computationally tractable. CG serve as an intermediate language for translating computer–oriented formalisms to and from natural languages. With a clear graphic representation, they serve as a readable —but formal— design and specification language. The next figure shows an example of a semantic net, that relates music bands, artists and basic data:

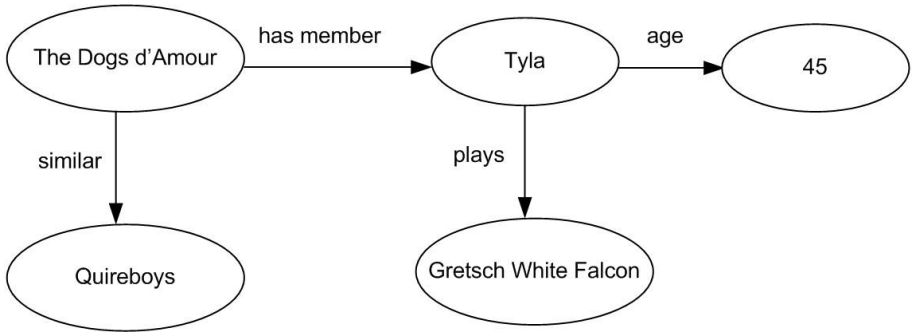

Figure 5.5: A semantic net.

Graph representation is a powerful tool for human understanding. However, in our context we need machine–processable representations.

RDF vocabulary allows to formally describe the previous example, and even serialize it using the XML language. RDF is, then, a data model for objects (resources) and the relations (properties) between them, and it provides simple semantics. A resource is an object, a thing we want to talk about. A resource has an URI (Universal Resoure Identifier). Properties are a special kind of resource that describe relations between resources (e.g: related with, age, plays, etc.). Properties are identified by URIs.

Statements assert the properties of resources. From a natural language point of view, a statement is composed by a *Subject–Predicate–Object* triple. From a more computer science point of view, this is equivalent to an *Object-Attribute-Value* triple, or in this context a *Resource-Property-Value* triple. A triple $[x, P, y]$ is equal to a logical formula $P(x, y)$, where the binary predicate P relates the object x to the object y. Values can be either resources or literals (e.g. strings).

A possible statement could be: "Oscar Celma is the owner of the web page http://foafing-the-music.iua.upf.edu".
This triple `["Oscar Celma", "http://www.mydomain.org/ontology#owner", "http://foafing-the-music.iua.upf.edu"]` is equal to the graph statement:

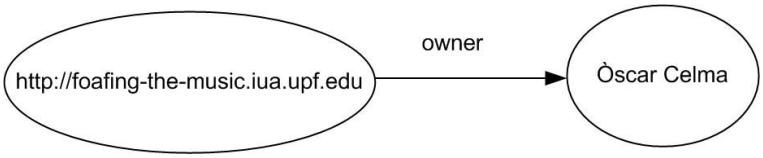

Figure 5.6: Graph representation of a triple.

It is a directed graph, where the nodes corresponds to the objects and the labelled arc is a property. The same statement can be represented in XML syntax (also known as RDF/XML):

```
<rdf:Description
  rdf:about="http://foafing-the-music.iua.upf.edu">
  <mydomain:owner>Oscar Celma</mydomain:owner>
</rdf:Description>
```

The `rdf:Description` makes a statement about the resource `http://foafing-the-music.iua.upf.edu`. The property (*owner*) is used as a tag within the description, and the value is the content of the tag[13].

[13]There are rules for creating abbreviated syntax of the statements. It is not goal of this Thesis to go further on these details.

Moreover, we can describe the person "Oscar Celma" by the resource with URL `http://www.mydomain.org/people/#44521`:

```
<rdf:Description
  rdf:about="http://www.mydomain.org/people/#44521">
  <mydomain:name>Oscar Celma</mydomain:name>
  <mydomain:title>Associate Professor</mydomain:title>
</rdf:Description>
```

In this case, the `rdf:Description` corresponds to two statements about the resource `http://www.mydomain.org/people/#44521` (the name, and the title of that person). Now, we can define a *course* that *is taught by* that resource:

```
<rdf:Description
  rdf:about="http://www.tecn.upf.es/~ocelma/edi2">
  <uni:courseName>Introduction to Databases</uni:courseName>
  <uni:creditsNumber>6</uni:creditsNumber>
  <uni:isTaughtBy rdf:resource="http://www.mydomain.org/people
     /#44521" />
</rdf:Description>
```

The resulting graph of the three previous examples is:

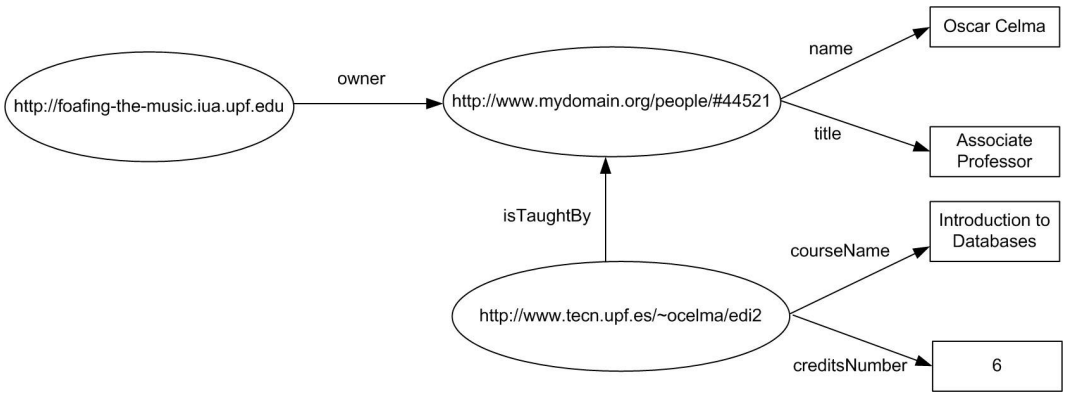

Figure 5.7: Graph representation of the previous RDF statements.

By now, we have defined a set of statements, but there still is no restrictions about them. For instance, we should state that the property *isTaughtBy* is only applied to courses (the subject) and professors (the object), or that an Associate Professor is a particular type of Professor, with some restrictions (maximum number of hours, needs to hold a PhD., etc). The RDF Schema vocabulary is intended to describe this information.

RDF Schema

RDF Schema (RDFS) is a vocabulary for describing properties and classes of RDF resources, and provides hierarchies of such properties and classes. RDFS vocabulary allows to define the semantics of the RDF statements.

As it is common in other disciplines, to describe a particular domain one can use classes and properties. RDFS provides mechanisms to define a particular domain using classes (and properties), hierarchies and inheritance. Classes model the entities (and their restrictions) of the domain, whereas properties provide relationships among the classes. Properties have a domain and range (similarly to mathematical functions), to impose restrictions on the values of the property.

Yet, there are some important missing features of RDFS:

- There are no local scope properties: `rdf:range` defines the range of a property for all classes. We cannot declare range restrictions that apply to some classes only
- There is no disjointness of classes
- Missing boolean combinations of classes: Union, Intersection, and complement
- No cardinality restrictions: restrictions on how many distinct values a property may or must take ("a person has two parents")
- No special characteristics of properties: transitive (greater than), unique (is mother of) and inverse (eats and is eaten by)

This limitations are solved in the OWL language, presented in the next section.

To conclude this section, a simile can be established among the existing technologies on the current Web, and the ones proposed by the Semantic Web community: while XHTML language makes the Web behave like a global book when viewed at the worldwide level, RDF and RDF Schema make it behave like a global database. Regarding the data structures, the basic RDF primitive is a directed graph, whereas XML representation is based on a tree. Thus, an RDF graph is on its own basically unrestricted and more powerful, in terms of expresiveness.

5.3.3 Ontology Vocabulary

An ontology is an explicit and formal specification of a conceptualization [Gru93]. In general, an ontology describes formally a domain of discurse. The requeriments for Ontology languages are: a well-defined syntax, a formal semantics, and a reasoning support that checks the consistency of the ontology, checks for unintended relationships between classes, and automatically classifies instances in classes.

The Web Ontology Language (OWL) has a richer vocabulary description language for describing properties and classes than RDFS. OWL has relations between classes, cardinality, equality, characteristics of properties and enumerated classes. The OWL language is build on RDF and RDFS, and uses RDF/XML syntax. OWL documents are, then, RDF documents.

The next example shows the definition of two classes:

```
<owl:Class rdf:ID="Singer">
  <rdfs:subClassOf rdf:resource="#Artist" />
</owl:Class>

<owl:Class rdf:ID="Song" />
```

Object property elements relate objects to other objects. For instance "a singer *sings* songs".

```
<owl:ObjectProperty rdf:ID="sings">
  <rdfs:domain rdf:resource="#Singer"/>
  <rdfs:range rdf:resource="#Song"/>
</owl:ObjectProperty>
```

Data type properties relate objects to datatype values. For example, the dataproperty that denotes the age of an *Artist*:

```
<owl:DataProperty rdf:ID="age">
 <rdfs:domain rdf:resource="#Artist"/>
 <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
<owl:DataProperty>
```

Property restrictions on classes are based on the use of `rdfs:subclassOf`. To say that class C satisfies certain conditions is equivalent to state that C is a subclass of C', where C' collects all objects that satisfy the conditions. For instance, a restriction on the kind of values the property can take:

```
<owl:Class rdf:about="#GuitarPlayer">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#plays"/>
```

```
   <owl:allValuesFrom rdf:resource="#Guitar"/>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

Or cardinality restrictions (a *music band* is composed by, at least, two members):

```
<owl:Class rdf:about="#Band">
  <rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasMember"/>
    <owl:minCardinality ="&xsd;nonNegativeInteger">
    2
    </owl:minCardinality>
  </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

OWL offers some special properties, such as: `owl:TransitiveProperty` (e.g. "has better grade than", "is taller than", "is ancestor of", etc.), `owl:SimmetricProperty` (e.g. "has same grade as", "is sibling of", etc.), `owl:FunctionalProperty` (a property that has almost one value for each object, e.g. "age"), and `owl:InverseFunctionalProperty` (a property for which two different objects cannot have the same value, e.g. "socialSecurityNumber"). For example, a *played with* property is simmetric:

```
<owl:ObjectProperty rdf:ID="playedWith">
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource="#Artist"/>
</owl:ObjectProperty>
```

There are three different OWL sublanguages. Each sublanguage offers a level of expressivity. **OWL Full** is the most expressive of the three sublanguages. It has no constraints. But, on the other hand the language becomes undecidable, so efficient reasoning is not guaranteed.

OWL DL is based on Description Logics. It has vocabulary partitioning, that is: any resource is allowed to be only a class, a data type, a data type property, an object property, an individual, a data value, or part of the built–in vocabulary. And, there is explicit typing in OWL DL, so the vocabulary partitioning must be stated explicitly. Property separation implies that the following can never be specified for data type properties: `owl:inverseOf`, `owl:FunctionalProperty`, `owl:InverseFunctionalProperty`, and `owl:SimmetricProperty`. And there

is a restriction for anonymous classes: they are only allowed to occur as the domain and range of either `owl:equivalentClass`, `owl:disjointWith` and as the range of `rdfs:subClassOf`.

Finally, **OWL Lite** has the same restrictions as OWL DL plus it is not allowed to use `owl:oneOf`, `owl:disjointWith`, `owl:UnionOf`, `owl:complementOf` nor `owl:hasValue`, Regarding cardinality statements: only values 0 and 1 are possible.

Appendix A shows a complete example that makes use of OWL to create an ontology of the music domain.

5.4 Moving Audiovisual descriptions to the Semantic Web

As explained in section 5.2, one of the main caveats of the MPEG-7 standard is the lack of formal semantics. Among other existing initiatives, the Semantic Web approach has defined a set of formal ontology languages to describe resources available on the Web. In this section we present a way to link the MPEG-7 standard within the Semantic Web technologies.

Chronologically, the first attempts to make MPEG-7 metadata semantics explicit where carried out, during the MPEG-7 standardisation process, by Jane Hunter [Hun99]. The proposal used RDF to formalise a small part of MPEG-7, and later incorporated some DAML+OIL[14] constructs to further detail their semantics [Hun01]. However, at that moment, there were not mature technologies for Web–wide metadata semantics formalisation. Moreover, XML had already a great momentum, so it was the logical choice to represent the MPEG-7 descriptors. From this point, once Semantic Web has matured, there have been more attempts to relate MPEG-7 with Web ontologies. However, none of them has retaken the initial effort to completely move MPEG-7 to the Semantic Web. This initiatives range from separated modules for existing MPEG-7 tools that offer reasoning capabilities for concretes aspects of multimedia management [DHL03], to a partial OWL modelling of the MPEG-7 Multimedia Description Schemes intended to facilitate MPEG-7 extensions [TPS04]. Moreover, they are not systematic; they are applied on an ad–hoc basis, what makes them very costly to

[14] A knowledge representation ontology previous to the OWL proposal

apply to the whole MPEG-7 standard.

The previous initiatives have produced very interesting results and are complementary to the objective of this section, i.e. to move the whole MPEG-7 to the Semantic Web. This way, we would have a core multimedia ontology that facilitates further extensions and reasoning capabilities, but also a complete semantics–aware solution for MPEG-7 metadata processing. The method we have used to perform is detailed in section 5.4.1. It is a generic XML Schema to OWL mapper combined with an XML to RDF translator.

The main caveat of semantic multimedia metadata is that it is sparse and expensive to produce [ONH03]. The initiatives presented in [Hun01], [Tro03], [DHL03] and [TPS04] are appropriate when applied to limited scopes. However, to increase the availability of semantic multimedia metadata and, in general, of semantic metadata, we need methods that are more productive. A direct solution is to take advantage from the large amount of metadata that has been already produced by the XML community.

There are many attempts to move metadata from the XML domain to the Semantic Web. Some of them just model the XML tree using the RDF primitives [Kle02]. Others concentrate on modelling the knowledge implicit in XML languages definitions, i.e. DTDs or the XML Schemas, using web ontology languages ([ABFS02][CXH04][HIMT03]). Finally, there are attempts to encode XML semantics integrating RDF into XML documents [LS03][PSP02]. However, none of them facilitates an extensive transfer of XML metadata to the Semantic Web in a general and transparent way. Their main problem is that the XML Schema implicit semantics are not made explicit when XML metadata instantiating this schemas is mapped. Therefore, they do not take profit from the XML semantics and produce RDF metadata almost as semantics–blind as the original XML. Or, on the other hand, they capture these semantics but they use additional ad–hoc semantic constructs that produce less transparent metadata.

5.4.1 Our proposal

In order to add semantics to MPEG-7 metadata we use an XML Schema to Web Ontology mapping and a transformation from XML instances to RDF semantic metadata. After that, we show that in the Semantic Web framework it is easier to integrate multimedia metadata coming from disparate sources and exploit the

implicit semantics for intelligent retrieval. Finally, once all metadata has been integrated, advanced ontologies and semantic rules are used to encode the necessary semantics to derive high–level concepts from content based descriptions.

We have chosen the XML Semantic Reuse methodology [Gar06]. It combines an XML Schema to web ontology mapping, called $XSD2OWL$, with a transparent mapping from XML to RDF, $XML2RDF$. The ontologies generated by $XSD2OWL$ are used during the XML to RDF step in order to generate semantic metadata that makes XML Schema semantics explicit.

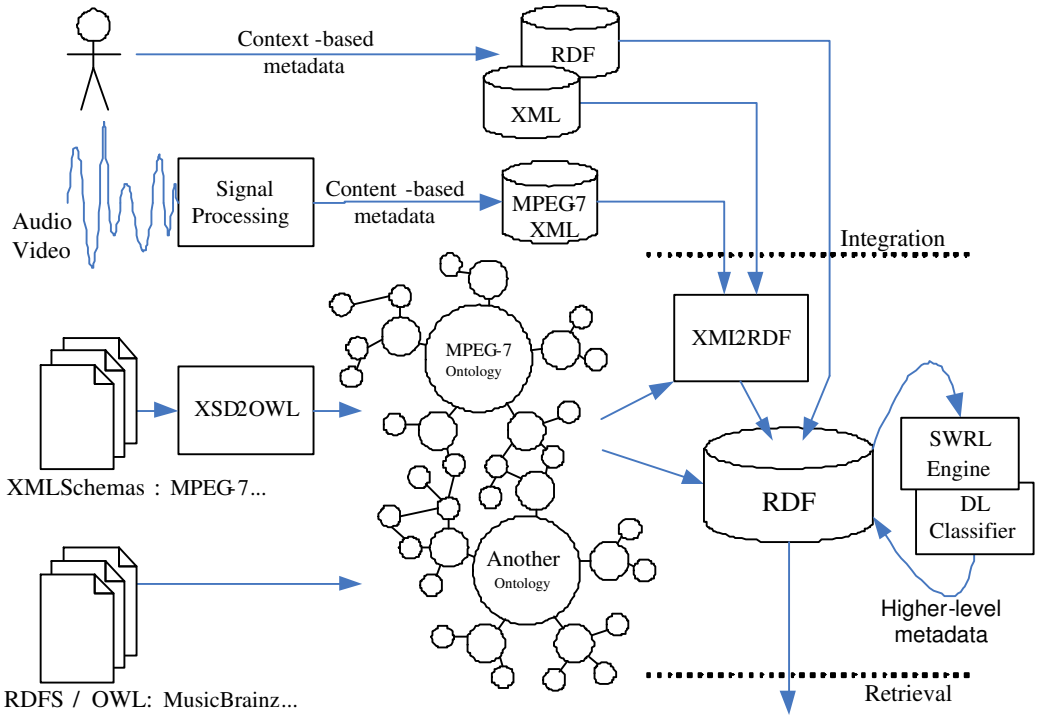

Figure 5.8: Metadata integration and retrieval architecture proposal.

Based on this XML world to Semantic Web domain mapping, we propose a system architecture that facilitates multimedia metadata integration and retrieval. The architecture is sketched in figure 5.8. The MPEG-7 OWL ontology, generated by $XSD2OWL$, constitutes the basic ontological framework for semantic multimedia metadata integration and appears at the centre of the architecture. Other ontologies and XML Schemas might be easily incorporated using the $XSD2OWL$ module. Semantic metadata can be directly fed into the system together with XML metadata, that is made semantic using the $XML2RDF$ module. XML MPEG-7 metadata has a great importance because it is commonly

used for (automatically extracted) low–level metadata that constitutes the basic input of the system.

This framework has the persistence support of a RDF store, where metadata and ontologies reside. Once all metadata has been put together, the semantic integration and retrieval of multimedia objects can take place.

Semantic Integration of Multimedia Metadata

The problem of integrating heterogeneous data sources has grown in importance within the last years. One of the main reasons is the increasing availability of web–based data sources. Even within a single organization, data from disparate sources must be integrated. Our approach to solve this problem is based on Web ontologies. As we focus on integration of multimedia assets, our base ontology is the MPEG-7 OWL ontology.

When multimedia metadata based on different schemes has to be integrated, the XML Schemas are first mapped to OWL. Once this first step has been done, these schemas are easily integrated into the ontological framework using OWL semantic relations for equivalence and inclusion: `subClassOf`, `subPropertyOf`, `equivalentClass`, `equivalentProperty`, `sameIndividualAs`, etc. These relationships capture the semantics of the data integration. Then, once metadata is incorporated into the system and semantically–decorated, the integration is automatically performed by applying inference. Our study on metadata integration is based on three different schemas: *MusicBrainz*[15] schema, *Foafing the Music* ontology[16] and a music vocabulary to describe performances[17]. *MusicBrainz* is a community music metadatabase that attempts to create a comprehensive music information site. The schema is written in RDF, and describes all the tracks, albums and artists available in their music repository. Their mappings to the MPEG-7 OWL ontology are shown in table 5.1.

Foafing the Music ontology describes content based descriptors extracted automatically from the audio itself, as well as some basic editorial information. The mappings of this schema to the MPEG-7 OWL ontology are summarized in table 5.2. An artist is defined as a subclass of the MPEG-7 Creator type, a track is defined as a subclass of the MPEG-7 AudioSegment and the audio Descriptor

[15] http://musicbrainz.org/mm/mm-2.1#
[16] http://foafing-the-music.iua.upf.edu/music-ontology#
[17] http:/www.kanzaki.com/ns/music#

$musicbrainz : Artist \subseteq mpeg7 : CreatorType$
$musicbrainz : Album \subseteq mpeg7 : CollectionType$
$musicbrainz : Track \subseteq mpeg7 : AudioSegmentType$
$dc : author \subseteq mpeg7 : Creator$
$dc : title \subseteq mpeg7 : Title$
$musicbrainz : sortName \subseteq mpeg7 : Name$
$musicbrainz : duration \equiv mpeg7 : MediaDuration$

Table 5.1: MusicBrainz to MPEG-7 OWL ontology mappings.

class describes the content-based properties of a track. This descriptor is linked with the MPEG-7 AudioDS type. Thus, all *Foafing the Music* descriptors' subclasses inherit the properties from the MPEG-7 Audio descriptor scheme. To characterize the descriptors related with the tonality of a song, *Foafing the Music* ontology defines some properties, such as mode and key. Finally, it defines rhythm descriptors to describe the rhythm component of a track, e.g. meter and tempo.

$foafingthemusic : Artist \subseteq mpeg7 : CreatorType$
$foafingthemusic : name \equiv mpeg7 : GivenName$
$foafingthemusic : Track \subseteq mpeg7 : AudioSegmentType$
$foafingthemusic : title \subseteq mpeg7 : Title$
$foafingthemusic : duration \equiv mpeg7 : MediaDuration$
$foafingthemusic : Descriptor \equiv mpeg7 : AudioDSType$
$foafingthemusic : mode \equiv mpeg7 : Scale$
$foafingthemusic : key \equiv mpeg7 : Key$
$foafingthemusic : tempo \equiv mpeg7 : Beat$
$foafingthemusic : meter \equiv mpeg7 : Meter$

Table 5.2: Foafing the Music ontology to MPEG-7 OWL ontology mappings.

The last of the three schemas, a music vocabulary to describe performances, is linked, as well with the MPEG-7 OWL (see table 5.3). This schema models —for example, in the classical music world— a concert with the conductor, performers, the whole program, time schedule, etc. The most general class related with a music piece is the *MusicalUnit*, from which all types of performances derived

(e.g. an opera performance, a symphony, a movement of the symphony, etc.). Decomposition of a musical unit is achieved by defining its sections, and we link it with the MPEG-7 AudioSegment. Finally, there is an Artist class, which all the agents of the performances (director, musician, singer, etc.) are subclass of. Therefore, we link the Artist class with MPEG-7 OWL and, automatically (transitivity property of `rdfs:subClassOf`) all subclasses are linked with the MPEG-7 OWL ontology.

$music : Music\_Unit \subseteq mpeg7 : AudioSegmentType$
$music : sections \equiv mpeg7 : AudioSegment$
$music : Artist \subseteq mpeg7 : CreatorType$
$music : key \equiv mpeg7 : Key$
$music : meter \equiv mpeg7 : Meter$

Table 5.3: Music Vocabulary ontology to MPEG-7 OWL ontology mappings.

Once these mappings are done, all the multimedia assets are integrated into the ontological framework; that is the MPEG-7 OWL linked with all the schemas. Now, querying the system for audio segments will retrieve information from all the different sources, transparently to the user.

Semantic Retrieval of Multimedia Metadata

Retrieving multimedia assets in the proposed architecture can be easily achieved by using semantic query languages like the *SPARQL* query language[18]. *SPARQL* can take profit from the implicit semantics. It can, as well, exploit the results of semantic rules for metadata integration in order to retrieve all the related multimedia information for a given query. In our case, *SPARQL* queries use the MPEG-7 OWL ontology vocabulary in order to integrate all data source. Using the mappings explained in the previous section, an *SPARQL* query can acquire information from *MusicBrainz*, *Foafing the Music*, the classical music ontology, etc.

A typical scenario that shows the usefulness of the architecture proposed could be the following: an Internet crawler is looking for audio data, and it downloads all the files. Getting editorial and related information for these audio files can be

[18]http://www.w3.org/TR/rdf-SPARQL-query/

achieved reading the information stored in the ID3 tag. Unfortunately, sometimes there is no basic editorial information like the title of the track, or the performer. However, content based descriptors can be computed for these files, including its MusicBrainz fingerprint, a string that uniquely identifies each audio file based on its content (improvements on how to calculate a robust fingerprint for an audio file are described in [CBKH02]). The next example shows an RDF/N3 description for a track with the calculated tempo and fingerprint:

```
<http://example.org/track#1> a foafingthemusic:Track;
 foafingthemusic:tempo "74";
 musicbrainz:puid "e3c41bc1-4fdc-4ccd-a471-243a0596518f".
```

Listing 5.4: Example of RDF/N3 description for a track (with the calculated tempo and fingerprint.)

On the other hand, MusicBrainz database has the editorial metadata —as well as the fingerprint already calculated— for more than 3 millions of tracks. For example, the RDF description of the song "Blowin' in the wind" composed by Bob Dylan:

```
<http://example.org/track#2> a musicbrainz:Track;
 dc:title "Blowin'␣in␣the␣wind";
 dc:author [musicbrainz:sortName "Bob␣Dylan"];
 musicbrainz:puid "e3c41bc1-4fdc-4ccd-a471-243a0596518f".
```

Listing 5.5: Example of RDF description of the song "Blowin' in the wind", composed by Bob Dylan.

A closer look to both examples 5.4 and 5.5, should highlight that the two resources are sharing the same MusicBrainz's fingerprint. Therefore, it is clear that, using a simple rule, one can assert that both audio files are actually the same file, that is to say the same instance in terms of OWL, owl:sameIndividualAs.

$$
\begin{aligned}
&mpeg7 : AudioType(track_1) \wedge mpeg7 : AudioType(track_2) \wedge \\
&musicbrainz : puid(track_1, puid_1) \wedge \\
&musicbrainz : puid(track_2, puid_2) \wedge (puid_1 = puid_2) \\
&\Rightarrow \\
&owl : sameIndividualAs(track_1, track_2)
\end{aligned}
$$

Figure 5.9: Simple rule to assert that two individual are the same.

From now on, we have merged the metadata from both sources and we have deduced that the metadata related with both tracks is, actually, referred to the same track. This data integration (at the instance level) is very powerful as

it can combine and merge context-based data (editorial, cultural, etc.) with content-based data (extracted from the audio itself). Finally, doing an *SPARQL* query that searches for all the songs composed by Bob Dylan that have at least a medium tempo (e.g beats-per-minute value greater or equal than 60), retrieves a list of songs, including "Blowin' in the wind". Moreover, there is no need for metadata provenance awareness at the end-user level. As the next example shows, all query terms are referred only to the MPEG-7 OWL ontology namespace:

```
PREFIX
mpeg7:<http://rhizomik.upf.edu/ontologies/2005/03/Mpeg7-2001.owl#
    >
SELECT ?title
WHERE
?track,<rdf:type>,mpeg7:AudioSegmentType .
(?track,<mpeg7:Title>,?title),(?track,<mpeg7:Creator>,?author) .
(?author,<mpeg7:Name>,"Bob␣Dylan"),(?track,<mpeg7:Beat>,?tempo) .
FILTER (?tempo >= 60)
ORDER BY (ASC(?title))
```

Listing 5.6: *SPARQL* expression to retrieving metadata among different schemas linked with MPEG-7 OWL.

Another interesting usage is the propagation of annotations. That is, when we have information from one source (i.e an audio file) and we want to propagate some of the annotations to another source. Given a song ($track_1$) with a set of high-level annotations (either supervised by a musicologist, or gathered through a process of web mining, for instance), and a song ($track_2$) that lacks some of these high-level descriptions, then we can apply a set of rules that can propagate part of the annotations of $track_1$ to $track_2$. To decide whether we can propagate this information, we need an extra component in the system that tell us how similar —based on automatically extracted audio features— are songs $track_1$ and $track_2$. If they are close together, then it makes sense to propagate some annotations from one song to another. Figure 5.10 exemplifies this case.

This annotation process could be supervised by an expert. Thus, the process of annotating would be, now, to check whether this propagated annotations make sense or not.

Based on this idea of propagating annotations, we have implemented a tool, named *Good Vibrations* [SACH06], that automatically propagates music user's annotations. *Good Vibrations* is a tool for music tagging, exploration and dis-

$mpeg7 : AudioType(track_1) \wedge mpeg7 : AudioType(track_2) \wedge$
$similars(track_1, track_2)$
$\Rightarrow$
$propagateAnnotations(track_1, track_2)$

Figure 5.10: Simple rule to propagate annotations from one song ($track_1$) to another ($track_2$).

covery. It is a Winamp[19] plugin that allows the quick "invention" of concepts and properties that can be tagged to songs. After a few hours of active tagging, the plugin starts automatically proposing the proper tags to the user, who is also allowed to correct them. The plugin generates playlists according to the user-defined concepts, and recommends related music either from the user's personal collection or from the Internet (through its connection to *Foafing the Music* system, presented in chapter 7). We emphasize that user intervention is crucial when adding semantics to the objects. Two users can *attach* different semantics to the same music object, and still both descriptions could be valid. Thus, we do not think that a pure bottom–up approach (i.e signal processing plus machine learning techniques) can alleviate the existing semantic gap. The process must be bidirectional, thus, combining bottom–up and top–down (driven by user) approaches, we believe that the semantic gap could be bridged.

5.5 Summary and Conclusions

This chapter has introduced the notions of multimedia ontologies, and the problems of annotating multimedia assets, integrating data from different sources, and retrieving audiovisual objects.

Section 5.2 has presented the MPEG-7 multimedia standard. Based on the limitations of the semantics expressiveness, section 5.4 has been guided by the need for a semantic multimedia metadata framework that facilitates multimedia applications development. It has been detected, as it is widely documented in the bibliography, that MPEG-7 is the biggest metadata framework created to date. MPEG-7 is based on XML Schemas and thus its metadata does not have a formal semantics. Consequently, there have been a lot attempts to move MPEG-7 to

[19] http://www.winamp.com

the Semantic Web. The approach presented is also in this direction, and uses a complete and automatic mapping of the whole MPEG-7 standard to OWL. It is based on a generic XML Schema to OWL mapping. It is important to note that this ontology is OWL-Full because the underlying XML Schema model has elements that might have complex and simple type values, i.e. object and data type in OWL terms. The previous mapping is complemented with an XML metadata instances to RDF mapping that completes a transparent transfer of metadata from the XML to the Semantic Web domain. Once in a semantic space, data integration, which is a crucial factor when several sources of information are available, is facilitated enormously.

5.5.1 Links with music recommendation

We have used the MPEG-7 OWL ontology as an upper-level multimedia ontology where three different music schemas have been linked. Thus, it is possible to retrieve related information from instances of all the sources. This is important for a music recommender that gathers information from different music collections, each one with a particular schema definition. With our approach, integration among several music collections is simple and straight–forward. Furthermore, detecting and merging instances from different sources permit to enhance the description of audio files, both content based and editorial data. This permits to a music recommender having more information about the songs.

High–level descriptors facilitate more accurate content retrieval and personalized recommendations. Thus, going one step beyond, it would be desirable to combine mid–level acoustic metadata with as much editorial and cultural metadata as possible. From this combination, more sophisticated inferences and rules would be possible. These rules derive hidden high–level metadata that could be, then, easily understandable by the end–user, enhancing user profiles. As an outline, figure 5.11 shows a simple rule that extracts a high–level descriptor from mid–level descriptors (i.e. a bootom-up approach) resulting from audio signal processing and mchine learning techniques. But again, user (or domain expert) intervention is needed, at least to validate that the induced rules makes sense.

This simple rule induces a possible value (e.g. *High*) for the danceability concept, based on acoustic (tempo and loudness) and editorial information (genre). The other way round to achieve a similar rule, is following a top–down approach,

$$
\begin{aligned}
&mpeg7 : AudioType(track) \wedge mpeg7 : Beat(track, t) \wedge \\
&(t > 120) \wedge mpeg7 : Loudness(track, l) \wedge \\
&(l > 0.9) \wedge mpeg7 : genre(track, Pop) \\
&\Rightarrow \\
&danceability(track, High)
\end{aligned}
$$

Figure 5.11: Simple rule to derive a value (High) for the danceability concept, from mid–level acoustic and editorial metadata.

with explicit creation of concepts by a user. Now, is the user who gives positive examples for a semantic label. From these instances, the system is able to predict possible values for the user–defined concepts. *Good Vibrations*, a plug–in for WinAmp, follows this approach. The plugin generates playlists according to the user–defined concepts, and recommends related music. Furthermore, the plugin starts automatically proposing the proper tags to the user, who is also allowed to correct them. This allows to semi–automatically enhance the descriptions of audio files, with semantic concepts created by the user himself.

Chapter 6

Prototype I: A music search engine

In this chapter we present the first of the two prototypes that we have implemented. The prototype, named *SearchSounds*, is a web–based music search engine that allows to discover music by means of content–based similarity.

This chapter is structured as follows: next section, 6.1, introduces the motivations and background of the implemented system. In section 6.2 we present the architecture of the system. Finally, the last section summarizes the work done and outlines the remaining work regarding the system.

6.1 Motivation

Nowadays, in the context of the World Wide Web, the increasing amount of available music makes very difficult, to the user, to find music he/she would like to listen to. To overcome this problem, there are some audio search engines[1] that can fit the user's needs. Some of the current existing search engines are nevertheless not fully exploited because their companies would have to deal with copyright infringing material. Music search engines have a crucial component:

[1]To mention a few (accessed on June, 1st, 2006):
http://search.singingfish.com/,
http://audio.search.yahoo.com/,
http://www.audiocrawler.com/,
http://www.alltheweb.com/?cat=mp3 and
http://www.altavista.com/audio/

an audio crawler, that scans the web and gathers related information about audio files [Kno04].

6.1.1 Syndication of Web Content

During the last years, syndication of web content —a section of a website made available for other sites to use— has become a common practice for websites. This originated with news and weblog sites, but nowadays is increasingly used to syndicate any kind of information. Since the beginning of 2003, a special type of weblog, named audio weblogs (or MP3 blogs), has become very popular. These blogs make music titles available for download. The music posted is explained by the blog author, and usually it has links that allow to buy the complete album or work. Sometimes, the music posted is hard to find or has not been issued in many years, and many MP3 blogs link strictly to music that is authorized for free distribution. In other cases, MP3 blogs include a disclaimer stating that they are willing to remove music if the copyright owner objects. Anyway, this source of semi-structured information is a jewel for web crawlers, as it contains the user's object of desire —e.g. an audio file—, and some textual information that is referring to the object.

The file format used to syndicate web content is XML. Web syndication is based on the RSS family and Atom formats. The RSS abbreviation is variously to refer to the following standards: Really Simple Syndication (RSS 2.0), Rich Site Summary (RSS 0.91 and 1.0) or RDF Site Summary (1.0).

Of special interest are the feeds that syndicate multimedia content. These feeds publish audiovisual information that is available on the net. An interesting example is the Media RSS (mRSS) specification[2], lead by *Yahoo!* and the multimedia RSS community. mRSS allows to syndicating multimedia files (audio, video, image) in RSS feeds, and adds several enhancements to RSS enclosures. Although mRSS is not yet widely used on the net, there are some websites that syndicates their multimedia content following the specification[3]. These feeds contain textual information, plus a link to the actual audiovisual file. As an example, listing 6.1 shows a partial RSS feed[4].

```
<rss version="2.0"
```

[2]http://search.yahoo.com/mrss/

[3]One of the most important ones is http://www.ourmedia.org

[4]Adapted from a real example in OurMedia website. http://www.ourmedia.org

```
xml:base="http://www.ourmedia.org"
xmlns:media="http://search.yahoo.com/mrss"
xmlns:dc="http://purl.org/dc/elements/1.1/"
>
<channel>
 <title>Example of a mRSS feed</title>
 <link>
  http://www.ourmedia.org/user/45801
 </link>
 <description>Recently published media
  items from Ourmedia.org</description>
 <language>en</language>
 <item>
  <title>Inside Track II</title>
  <link>
   http://www.ourmedia.org/node/...
  </link>
  <description>Rock music with a funky
   beat and electric lead guitar riffs (...)
  </description>
  <pubDate>
   Mon, 13 Feb 2006 01:35:49 -0500
  </pubDate>
  <dc:creator>
   Bill Brettschneider
  </dc:creator>
  <category domain="urn:ourmedia:term:35">
   Alternative Rock
  </category>
  <category domain="urn:ourmedia:term:582">
   funk
  </category>
  <category domain="urn:ourmedia:term:727">
   guitar
  </category>
  <enclosure
   url="http://archive.org/.../file.mp3"
   length="3234212"
   type="application/octet-stream" />
 </item>
 <item>
  <title>Another item</title>
  ...
 </item>
</channel>
</rss>
```

Listing 6.1: Example of a media RSS feed.

The example shows an item with all its information: the title of the item, the description, the publication date, the editor of the entry, and a set of categories (similar to tags, but controlled from a given taxonomy). *SearchSounds* mines

this information in order to retrieve relevant audio files based on text queries.

6.2 System overview

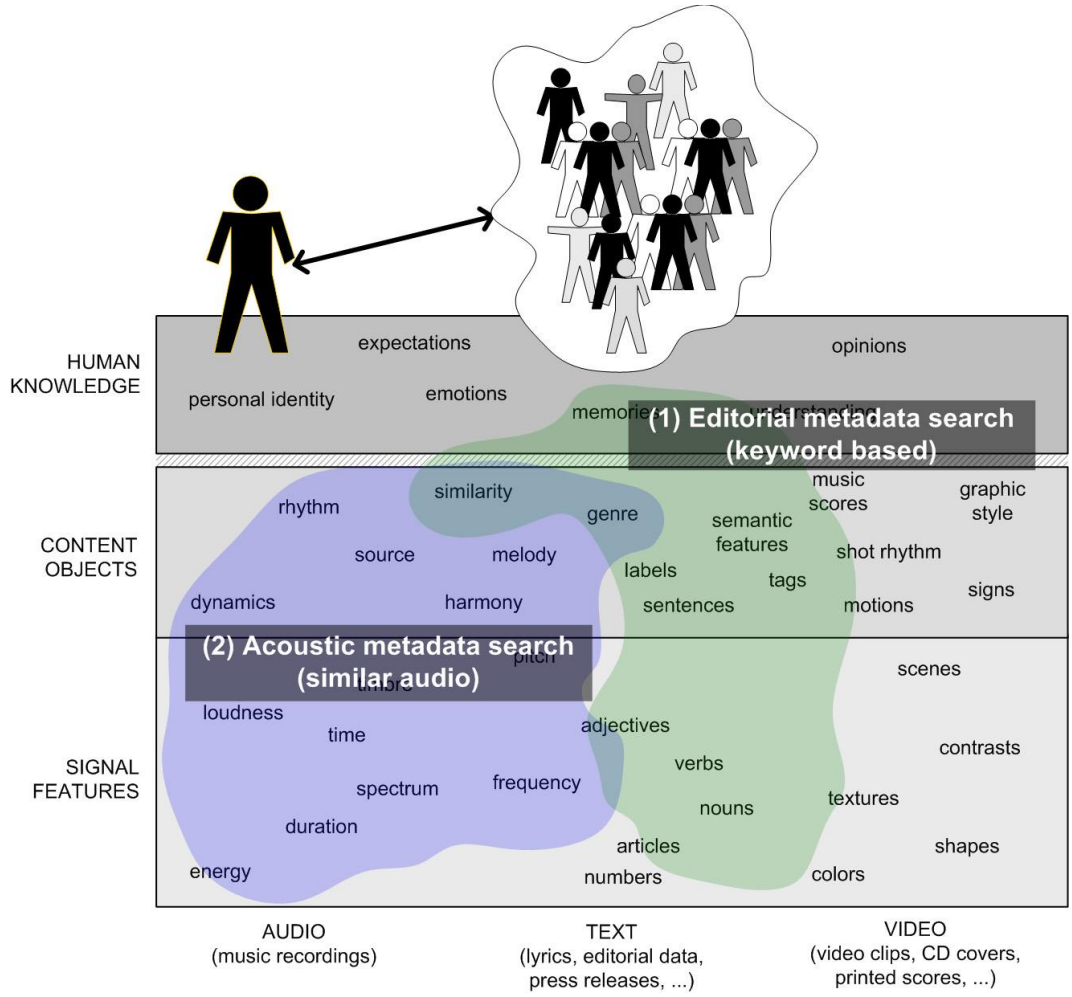

Figure 6.1: *SearchSounds* makes use of editorial, cultural and acoustic metadata. The system retrieves audio files from a user's query *(1)*, and each track has a list of (content–based) similar titles *(2)*.

SearchSounds exploits and mines all the music related information available from MP3 weblogs. The system gathers editorial, cultural, and acoustic information from the crawled audio files. The input of the system is a query composed by text keywords. From these keywords, the system is able to retrieve a list of audio files related with the query. Each audio file provides a link to the original weblog, and a list of similar titles. This similarity is computed by means

of content–based audio description. Thus, from the results of a text query, a user can discover related music by navigating onto the audio similarity plane. Figure 6.1 shows the relationship between the music information plane and the metadata that *SearchSounds* uses.

It is worth to mention that there is no user profiling or any kind of user representation stored in the system. This is a limitation, as the system does not make any personalized recommendations. However, this limitation is solved in the next prototype (explained in chapter 7). The main components of the system are the audio crawler and the audio retrieval system. Figure 6.2 depicts the architecture of the system.

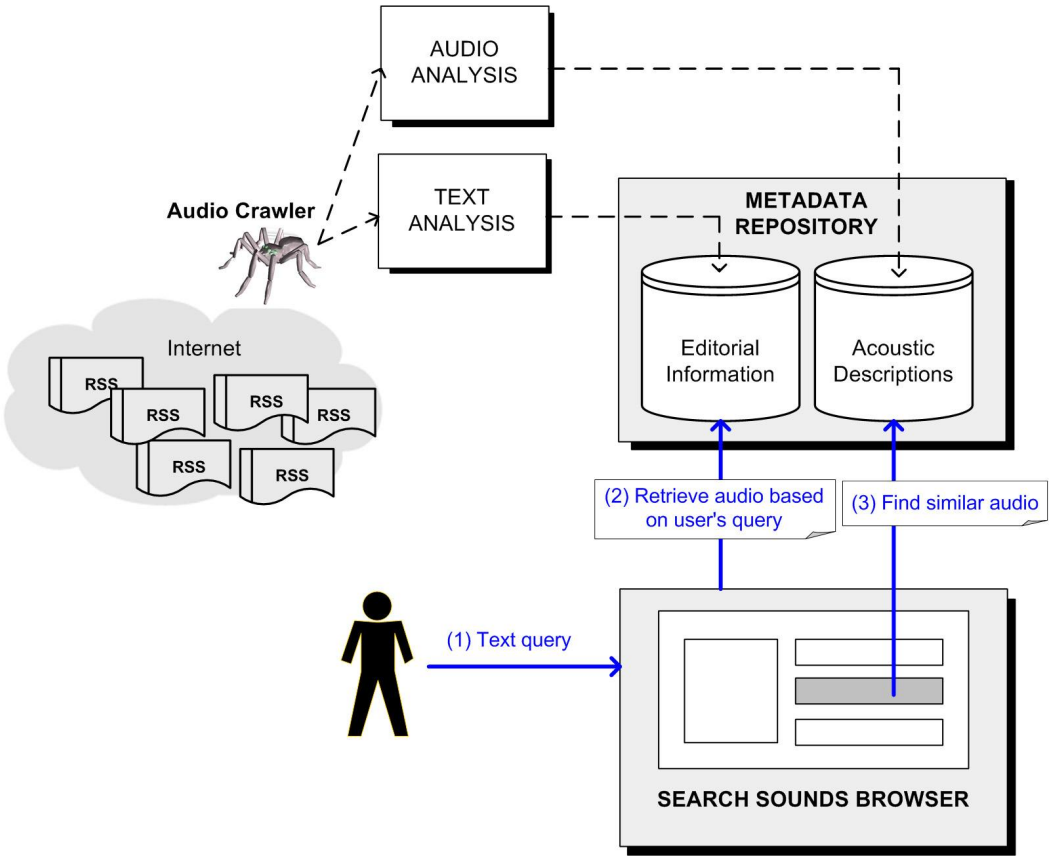

Figure 6.2: *SearchSounds* architecture. The main components are the audio crawler, and the audio retrieval system.

6.2.1 Audio Crawler

The system has an audio spider module that crawls the web. All the gathered information is stored into a relational database. The audio crawler starts the process from a manually selected list of RSS links (that point to MP3–blogs). Each RSS file contains a list of entries (or *items*) that link to audio files. The crawler seeks for new incoming items —using the *pubDate* item value and comparing with the latest entry in the database— and stores the new information into the database. Thus, the audio crawler system has an historic information of all the items that appeared in a feed.

From the previous RSS example (see example 6.1, presented in section 6.1.1), the audio crawler stores the text from the *title*, the content of the *description*, the assigned terms from the taxonomy (*category* tags), and the link to the audio file (extracted from the *enclosure url* attribute).

6.2.2 Audio Retrieval System

The logical view of a crawled feed item can be described by the bag–of–words approach: a document is represented as a number of unique words, with a weight (in our case, the tf/idf function) assigned to each word [BYRN99]. Special weights are assigned to the music related terms, as well as the metadata (e.g ID3 tags) extracted from the audio file. Similar to our approach, [VB04] presents a proposal of modifying the weights of the terms pertaining to the musical domain.

Moreover, basic natural language processing methods are applied to reduce the size of the item description (elimination of stopwords, and apply Porter's stemming algorithm [Por80]). The information retrieval (IR) model used is the classic vector model approach, where a given document is represented as a vector in a multidimensional space of words (each word of the vocabulary is a coordinate in the space).

Full text search

The similarity function, $sim(d_j, q)$, between a query (q) and a document (d_j) is based on the tf/idf weighting function (already presented in section 3.2.2). In this case, the similarity function between the query and a document used is:

$$sim(d_j, q) \sim \sum_{t \in q} \frac{TF_{t,j}}{|\vec{d_j}|} \cdot IDF_t \tag{6.1}$$

We use a simplified version of the classic cosine similarity, because a search engine only cares about the ranking order of the results —and do not care about the similarity value itself.

This IR model is well suited not only for querying via artists' or songs' names, but for more complex text queries such as: *"funky guitar riffs"* or *"traditional Irish tunes"*.

The retrieval system outputs the documents (i.e. feed entries) that are relevant to the user's query, ranked by the similarity function. Figure 6.3 depicts the retrieved audio files for *"traditional Irish music"* query.

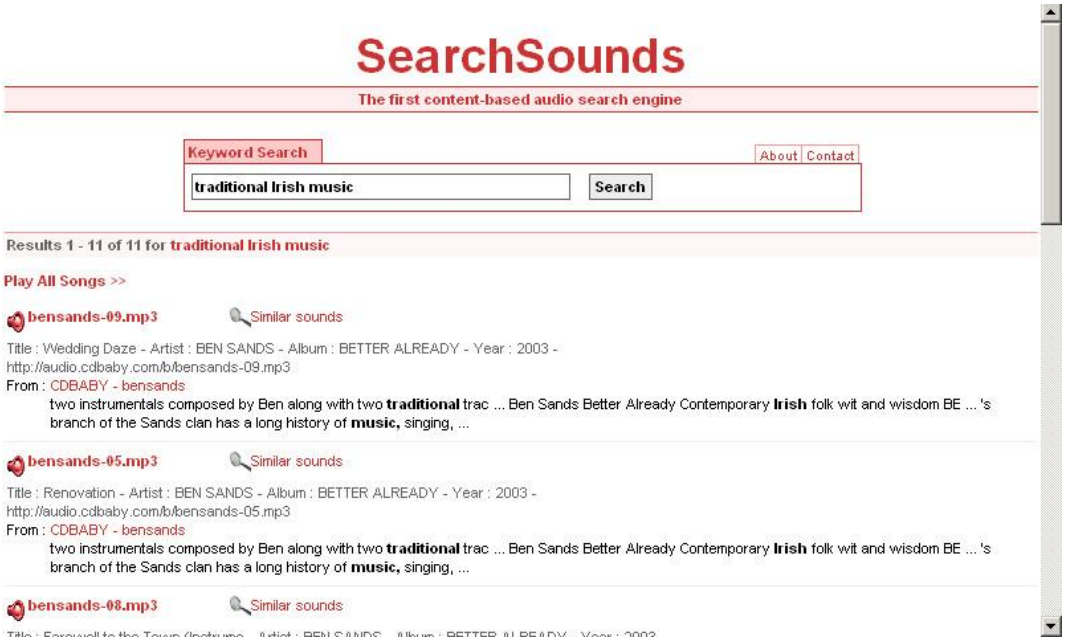

Figure 6.3: *SearchSounds* screenshot. It shows the first results from *"traditional Irish music"* query.

Content based similarity

Based on the results obtained from the user's textual query, the system allows to find similar audio files by means of content–based audio similarity. Each link to an audio file has a *"Find similar"* button that retrieves the most similar audio files, based on a set of low and mid-level audio descriptors. These descriptors are

extracted from the audio and represent properties such as: rhythm, harmony, timbre and instrumentation, intensity, structure and complexity [CKW+05].

This exploration (or browsing) mode allows to the user to discover music —related to his original (text based) query— that would be more difficult to discover by using textual queries only. To our knowledge, nowadays, this is the only web–based audio search engine that allows this type of content–based navigation —for audio files crawled from Internet. There is an analogy between this type of navigation and, for example, Google's "find web pages that are similar to a given HTML page". In our case, similarity among items are based on audio similarity, whereas Google approach is based on the textual content of the HTML page. Still, both browsing approaches are based on the *content* analysis of the retrieved object.

Figure 6.4 depicts a list of similar audio files, from a file obtained via the text query *"traditional Irish music"*.

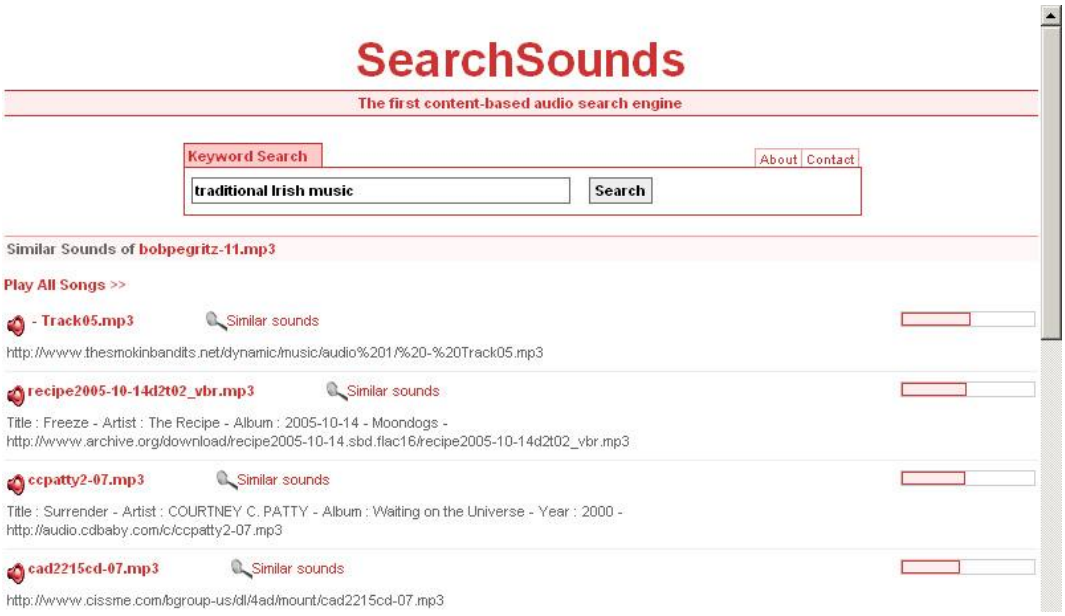

Figure 6.4: *SearchSounds* screenshot for music discovery by means of audio similarity.

6.3 Summary and Conclusions

We have presented an audio crawler focused on MP3 weblogs. Out of the crawling process, each feed item is represented as a text document, containing the item content, as well as the links to the audio files. Then, a classic text retrieval system outputs relevant items related to the user's query. Moreover, a content-based navigation allows to browse among the retrieved items and discover new music and artists by means of audio similarity.

Some future remaining tasks include: a experiment addressing the interaction between content–based and text–based querying, and how users differently employ them. Although, this experiment needs to analyze usage data of the system and, currently, we do not have such data. Finally, a relevance feedback method to tune the system and get more accurate results (specially for the content–based navigation) should be taken into account. The system is available at **http://www.searchsounds.net**.

Chapter 7

Prototype II: A hybrid music recommender

This chapter presents the second of the two prototypes developed. It is a music recommender system, named *Foafing the Music*, that allows to discover a wide range of music based on user profiles. Moreover, the system exploits music related information that is being syndicated on websites. From the gathered data, the system is able to filter and recommend it to the user, according to his profile.

This chapter is structured as follows: next section 7.1 outlines the motivation, and existing related systems. Then, section 7.2 presents the system architecture. Finally, in section 7.3, we end up with a brief summary and the remaining tasks to be done.

7.1 Motivation

The World Wide Web has become the host and distribution channel of a broad variety of digital multimedia assets. Although the Internet infrastructure allows simple straightforward acquisition, the value of these resources lacks of powerful content management, retrieval and visualization tools. Music content is no exception: although there is a sizeable amount of text-based information about music (album reviews, artist biographies, etc.) this information is hardly associated to the objects they refer to, that is music music files (MIDI and/or audio). Moreover, music is an important vehicle for communicating other people something relevant about our personality, history, etc.

In the context of the Semantic Web, there is a clear interest to create a Web of machine-readable homepages describing people, the links among them, and the things they create and do. The FOAF (*Friend Of A Friend*) project[1] provides conventions and a language "to tell" a machine the sort of things that a user says about herself in her homepage. FOAF is based on the RDF/XML[2] vocabulary. We can foresee that with the user's FOAF profile, a system would get a better representation of the user's musical needs. On the other hand, the RSS vocabulary[3] allows to syndicate Web content on Internet. Syndicated content includes data such as news, events listings, headlines, project updates, as well as music related information, such as new music releases, album reviews, podcast sessions, incoming gigs, etc.

7.1.1 Related systems

Most of the current music recommenders are based on collaborative filtering approach (see 3.2.2). Examples of such systems are: *Last.fm*, *MyStrands*, *MusicMobs*[4], *Goombah Emergent Music*[5], *iRate*[6], and *inDiscover*. The basic idea of a music recommender system based on collaborative filtering is:

1. To keep track of which artists (and songs) a user listens to —through iTunes, WinAmp, Amarok, XMMS, etc. plugins,
2. To search for other users with similar tastes, and
3. To recommend artists (or songs) to the user, according to these similar listeners' taste.

On the other hand, the most noticeable system using (manual) content based descriptions to recommend music is *Pandora*. The main problem of the system is the scalability, because all the music annotation process is done manually.

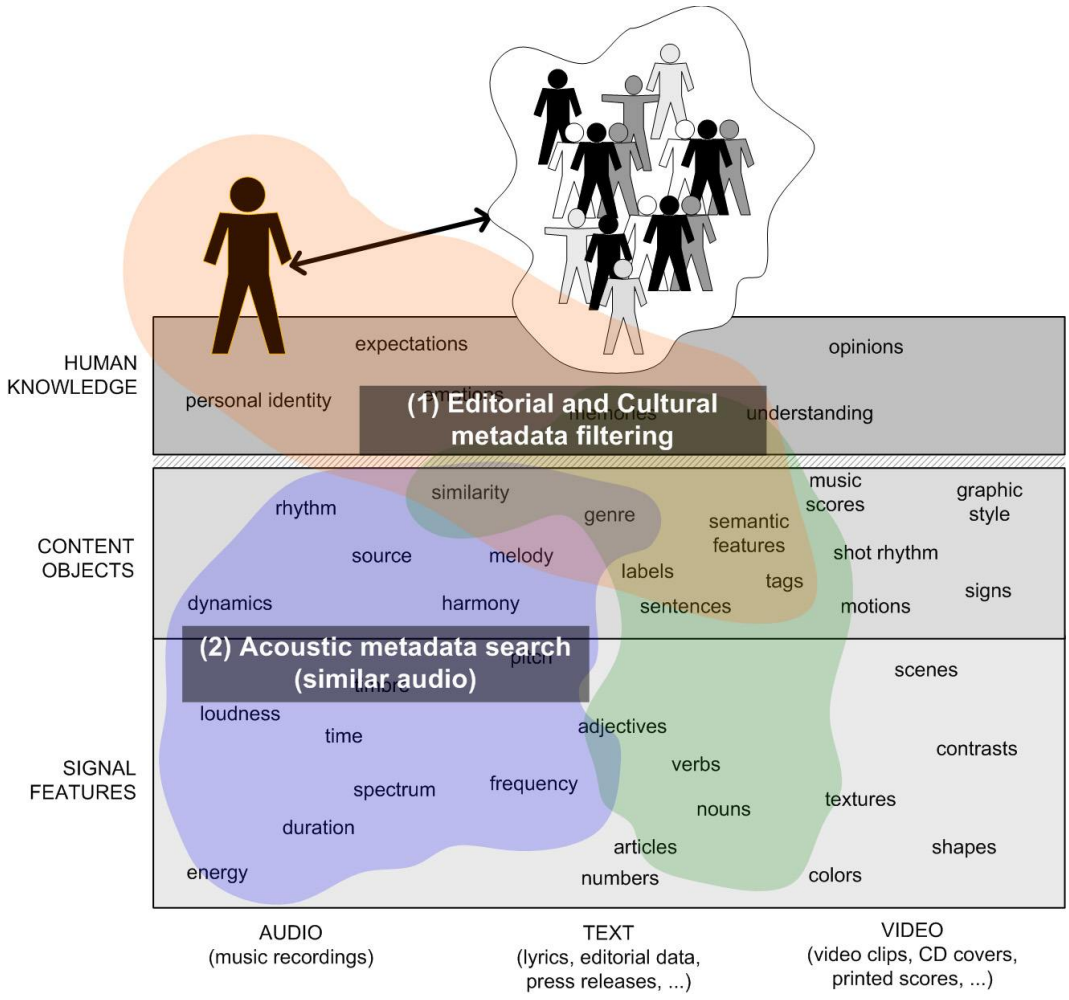

Figure 7.1: *Foafing the Music* and the music information plane.

7.2 System overview

The main goal of the *Foafing the Music* system is to recommend, to discover and to explore music content; based on user profiling (via FOAF descriptions), context based information (extracted from music related RSS feeds), and content based descriptions (automatically extracted from the audio itself). All of that being based on a common ontology that describes the musical domain. Figure 7.1 shows the relationship between the music information plane, and the different

[1]http://www.foaf-project.org
[2]http://www.w3.org/RDF
[3]http://web.resource.org/rss/1.0/
[4]http://www.musicmobs.com
[5]http://goombah.emergentmusic.com/
[6]http://irate.sourceforge.net

sources of metadata that the system exploits. Compared to the first prototype (*SearchSounds*), *Foafing the Music* holds a user profile representation, based on the FOAF initiative (already presented in section 3.2.1). A FOAF user profile allows to filter music related information according to user's preferences.

To our knowledge, nowadays it does not exist any system that recommends items to a user, based on her FOAF profile. Anyway, there is the *FilmTrust* system[7]. It is a part of a research study aimed to understanding how social preferences might help web sites to present information in a more useful way [GP05]. The system collects user reviews and ratings about movies, and holds them into the user's FOAF profile.

The overview of the system is depicted in figure 7.2. The next two sections explain the main components of the system, that is how to gather data from third party sources, and how to recommend music to the user.

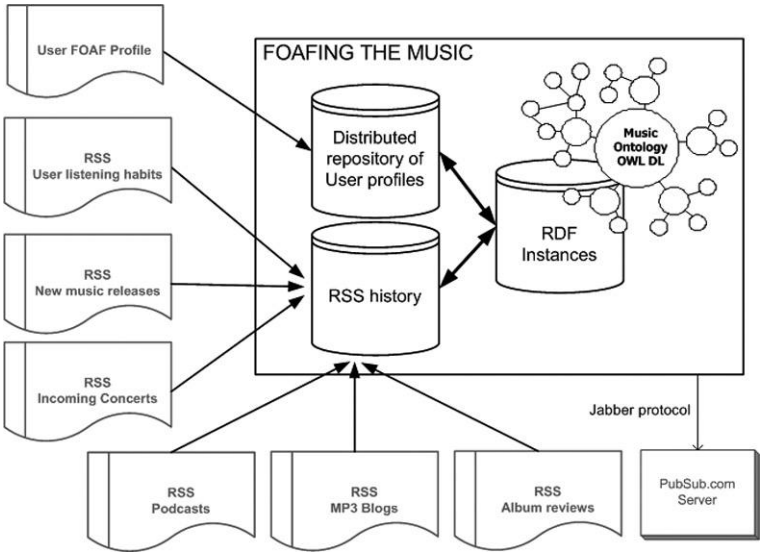

Figure 7.2: Architecture of the *Foafing the Music* system.

7.2.1 Gathering information

Personalized services can raise privacy concerns, due to the acquisition, storage and application of sensitive personal information [PdRME04]. A novelty approach is used in our system: information about the users is not stored into the system in any way. User's profiles are based on the FOAF initiative, and the

[7]http://trust.mindswap.org/FilmTrust

system has only a link pointing to the user's FOAF URL. Thus, the sensitivity of this data is up to the user, not to the system. Users' profiles of the system are distributed over the net.

Regarding music related information, *Foafing the Music* exploits the mashup approach. The system uses a set of public available APIs and web services sourced from third party websites. These information can come in any of the different RSS family (v2.0, v1.0, v0.92 and mRSS), as well as in the Atom format. Thus, the system has to deal with syntactically and structurally heterogeneous data. Moreover, the system keeps track of all the new items that are published in the feeds, and stores the new incoming data into a historic relational database. Input data of the system is based on the following information sources:

- **User listening habits**. To keep track of the user's listening habits, the system uses the services provided by Last.fm. This system offers an API, as well as an RSS feedthat provides the most recent tracks a user has played. Each item feed includes, then, the artist name, the song title and a timestamp (indicating when the user has listened to the track).

- **New music releases**. The system uses a set of RSS that notifies new music releases. Next table shows the contribution of each RSS feed into the historic database of the system:

RSS Source	Percent
iTunes	45.67%
Amazon	42.33%
Oldies.com	2.92%
Yahoo Shopping	0.29%
Others	8.79%

- **Upcoming concerts**. The system uses a set of RSS feeds that syndicates music related events. The websites are: Eventful.com, Upcoming.org, San Diego Reader[8] and SubPop record label[9]. Once the system has gathered all the new items, it queries to the Google Maps API (v2.0) to get the geographic location of the venues.

[8]http://www.sdreader.com/
[9]http://www.subpop.com/

- **Podcast sessions**. The system gathers information from a set of RSS feeds that publish podcasts sessions.

- **MP3 Blogs**. The system gathers information from a list of MP3 blogs that talk about artists and its songs. Each item feed contains a list of links to the audio files (see Table 7.1 for more detailed information).

- **Album reviews**. Information about album reviews are crawled from the RSS published by Rateyourmusic.com, Pitchforkmedia.com, and Rolling Stone online magazine[10].

Table 7.1 shows some basic statistics of the data that has been gathered since mid April, 2005 until the first week of July, 2006 (except for the album reviews that started in mid June, 2005). These numbers show that the system has to deal with a daily fresh incoming data.

RSS Source	# Seed feeds	# Items crawled per week	# Items stored
New releases	44	980	58,850
Concerts	14	470	28,112
Podcasts	830	575	34,535
MP3 blogs	86	2486 (avg. of 19 audios per item)	149,161
Reviews	8	458	23,374

Table 7.1: Information gathered from RSS feeds is stored into a historic relational database.

On the other hand, we have defined a music ontology[11] (OWL DL) that describes basic properties of the artists and the music titles, as well as some descriptors extracted from the audio (e.g. key, mode, tempo, etc.). In [GC05] we propose a way to map our ontology and the MusicBrainz ontology, within the MPEG-7 standard, that acts as an upper-ontology for multimedia description.

A focused web crawler has been implemented in order to add instances to the music ontology. The crawler extracts metadata of artists and songs, and the relationships between artists (such as: "related to", "influenced by", "followers of", etc.). The seed sites to start the crawling process are music metadata

[10]http://www.rollingstone.com/

[11]The OWL DL music ontology is available at: http://foafing-the-music.iua.upf.edu/music-ontology#

providers[12], and independent music labels[13]. Thus, the music repository does not consist only of mainstream artists.

Based on the music ontology (see Appendix A), the example 7.1 shows the RDF/XML description of an artist from Garageband.com.

```
<rdf:Description rdf:about="http://www.garageband.com/artist/
   randycoleman">
 <rdf:type rdf:resource="&music;Artist"/>
 <music:name>Randy Coleman</music:name>
 <music:decade>1990</music:decade>
 <music:decade>2000</music:decade>
 <music:genre>Pop</music:genre>
 <music:city>Los Angeles</music:city>
 <music:nationality>US</music:nationality>
 <geo:Point>
   <geo:lat>34.052</geo:lat>
   <geo:long>-118.243</geo:long>
 </geo:Point>
 <music:influencedBy
     rdf:resource="http://www.coldplay.com"/>
 <music:influencedBy
     rdf:resource="http://www.jeffbuckley.com"/>
 <music:influencedBy
     rdf:resource="http://www.radiohead.com"/>
</rdf:Description>
```

Listing 7.1: Example of an artist individual

Example 7.2 shows the description of a track individual of the above artist:

```
<rdf:Description rdf:about="http://www.garageband.com/song?|pe1|
   S8LTM0LdsaSkaFeyYG0">
 <rdf:type rdf:resource="&music;Track"/>
 <music:title>Last Salutation</music:title>
 <music:playedBy rdf:resource="http://www.garageband.com/artist/
    randycoleman" />
 <music:duration>4.07</music:duration>
 <music:key>D</music:key>
 <music:keyMode>Major</music:keyMode>
 <music:tonalness>0.84</music:tonalness>
 <music:tempo>72</music:tempo>
</rdf:Description>
```

Listing 7.2: Example of a track individual

These instances are used in the recommendation process, to recommend related artists based on a user's profile. Next section explains the music recom-

[12]Such as http://www.mp3.com, http://music.yahoo.com, http://www.rockdetector.com, etc.

[13]E.g. http://www.magnatune.com, http://www.cdbaby.com and http://www.garageband.com

mendation process.

7.2.2 Music Recomendation process

Music recommendations, in the *Foafing the Music* system, are generated according to the following steps:

1. Get music related information from user's FOAF interests, and user's listening habits
2. Detect artists and bands
3. Compute similar artists, and
4. Rate results by relevance.

In order to gather music related information from a FOAF profile, the system extracts the information from the FOAF interest property (if dc:title is given then it gets the text, otherwise it gathers the text from the title tag of the resource).

Based on the music related information gathered from the user's profile and listening habits, the system detects the artists and bands that the user is interested in (by doing a SPARQL query to the artists' individuals repository). Once the user's artists have been detected, artist similarity is computed. This process is achieved by exploiting the RDF graph of artists' relationships.

The system offers two ways of recommending music information. *Static* recommendations are based on the favourite artists encountered in the FOAF profile. We assume that a FOAF profile would be barely updated or modified. On the other hand, *dynamic* recommendations are based on user's listening habits, which is updated much more often that the user's profile. With this approach the user can discover a wide range of new music and artists.

Once the recommended artists have been computed, *Foafing the Music* filters music related information coming from the gathered information (see section 7.2.1) in order to:

- Get new music releases from *iTunes*, *Amazon*, *Yahoo Shopping*, etc.
- Download (or stream) audio from MP3–blogs and Podcast sessions,

- Create, automatically, XSPF[14] playlists based on audio similarity,
- Read Artists' related news, via the *PubSub* server[15]
- View upcoming gigs happening near to the user's location, and
- Read album reviews.

Syndication of the website content is done via an RSS 1.0 feed. For most of the above mentioned functionalities, there is a feed subscription option to get the results in the RSS format.

7.2.3 Implementation details

The system is implemented in PHP. The parsing of the FOAF profiles uses the RAP library (RDF API for PHP)[16]. To query the music ontology, the system makes use of the SPARQL language.

Foafing the Music uses the third party server *PubSub.com* to get artists' related news. *PubSub* is a matching service that instantly notifies a user whenever new content matching user's subscription is created. *Foafing the Music* can dynamically create a subscription that contains the favourite artists of the user. The communication between both systems is done via the Jabber protocol. To interact with the *PubSub* server a PHP class has been implemented. This class allows to connect to the server, to authenticate, and to create/retrieve a subscription. The output of the *PubSub* is displayed into the system as an RSS feed.

7.3 Summary and Conclusions

We have proposed a system that filters music related information, based on a given user's profile and user's listening habits. A system based on FOAF profiles and user's listening habits allows to "understand" a user in two complementary ways; psychological factors —personality, demographic preferences, socio-economics, situation— and explicit musical preferences. In the music field

[14]http://www.xspf.org/. XSPF is playlist format based on XML syntax
[15]http://www.pubsub.com
[16]http://www.wiwiss.fu-berlin.de/suhl/bizer/rdfapi/

context, we expect that filtering information about new music releases, artists' interviews, album reviews, etc. can improve a recommendation system in a dynamic way.

Foafing the Music is accessible through **http://foafing-the-music.iua.upf.edu**

Chapter 8

Conclusions and Future Work

This chapter presents the conclusions of the work presented, as well as the remaining work to conclude the PhD. Next section summarizes the first contributions of the author. Then, a proposal of the work to be done is outlined in section 8.2.

8.1 Summary of Contributions

This Thesis makes a number of contributions. We recap them in the following summary:

Formalization of the Music Recommendation Problem. We have analysed the music recommendation problem, as an instanciation of the *general* recommendation problem. We have presented the current methods (and related work) used to recommend music assets, as well as reviewed some proposals of modelling user preferences in the music field.

Definition of the Music Information Plane for describing music objects. We have proposed the music information plane as a way to express the different facets of music knowledge management (based on editorial, cultural and acoustic metadata). This holistic approach has led us to describe complex components of the music objects. Furthermore, these descriptions allow to enhance and improve music recommendation systems.

An ontological framework for semantic integration and retrieval of audiovisual metadata. We have proposed a framework, based on an OWL version of the MPEG-7 standard, that allows to integrate and retrieve metadata from different audiovisual repositories. As a testbed example, we have integrated

three different music schemas (*MusicBrainz*, *Foafing the Music*, and a Music Vocabulary to describe performances) with the MPEG-7 OWL ontology. After that, semantic retrieval of music information can be achieved independently of the original music repository. Moreover, we have presented a concrete scenario of the framework, that is the (semi–automatic) propagation of annotations, as a way to enhance and extend the metadata of the music content.

A music search engine and discovery prototype. We have implemented a music search engine, named *SearchSounds*, that exploits content–based descriptions as a way to explore and discover music. The system crawls information coming from RSS feeds of MP3–blogs. The novelty of the system is that it can retrieve similar titles (available from Internet) based on a user's text query.

A hybrid music recommender prototype. We have implemented a music recommendation system, named *Foafing the Music*, based on the ontological framework proposed. The system is inspired by the ideas of the Semantic Web initiative, and it does use the FOAF notation for modelling user preferences. Web content crawled from music related sites are filtering according to the user profile. Yet, the system does not exploit all the possibilities of the proposed semantic framework.

8.2 Future work

In this section we describe a detailed road map of the planned future work research, and we look at the tasks that need to be covered by the PhD thesis. One of the most important remaining tasks is the evaluation and comparison of the models and algorithms presented.

Evaluation strategies

One important aspect of any research involving the development of models or algorithms is the evaluation of the achievements. This evaluation can be seen under two different aspects. The first one involves an experiment with subjects that are evaluating the models or the algorithms' output. In this case, the goal is to proof that the application of the model or the algorithm yields a noticeable benefit in the desired context. The second aspect includes the proof validation, or the direct validation of the models' prediction against a reliable ground–truth.

Subjective evaluation of playlists

In our case, we are planning to carry on two different evaluations. The first one will measure the subjective qualitative evaluation of automatically generated playlists. The planning is to create different playlists based on three methods: collaborative–filtering, content–based, and a hybrid method. The seed artists (or genres) to create the playlists will be selected by the user. Thus, based on the selection done, the user will rate each playlist. The output of the evaluation will assess which playlist fits better the user expectations, based on a predefined criterion. The data to create the playlists will come from the following sources: for CF we will use the artist relationships available at last.fm, CB similarity will be based on the algorithms developed by the MTG[1], and the hybrid method will use the data of the *Foafing the Music* prototype, that merges content–based similarity and artists' relationships crawled from specialized websites.

Analysis of the Topology of Music Recommendation Networks

The second evaluation is based on the topology or structure of the recommendationd methods. We are planning to do an analysis of the recommendation networks at the song level (each song is a node, and songs are linked by their resemblance). Previous work done by the author has been focused on studying the topology of several music recommendation networks, which arise from relationships between artists [CCKMB06]. The analysis uncovers the emergence of complex network phenomena in these kinds of recommendation networks, built considering artists as nodes and their resemblance as links. We observe structural properties that provide some hints on navigation and possible optimizations on the design of music recommendation systems. Finally, the analysis derived from existing music knowledge sources provides a deeper understanding of the human music similarity perception.

The experiment to carry on is the evaluation and comparison of the topology of two recommendation networks at the song level, created with CF and CB methods. We are currently gathering song similarity data from *Yahoo! Music* website. They have computed song similarity based on user's listening habits from the *Y! LaunchCast* streaming radio station. Regarding CB data, we already have computed audio similarity of more than 1 million songs. The experiment

[1]See the *MusicSurfer* prototype for a glimpse, at http://musicsurfer.iua.upf.edu

consists on evaluating the complex network fenomena, and comparing both networks. Furthermore, the analysis of these two networks can arise interesting results about the inherent topology of each recommendation approach.

Improving the prototypes

There are some pending work regarding the prototypes presented. Some future tasks of the *SearchSounds* system include: a experiment addressing the interaction between content–based and text–based querying, and how users differently employ them. Although, this experiment needs to analyze usage data of the system and, currently, we do not have such data. It is important, too, to add a relevance feedback method to improve the system, and get more accurate results from the content–based similarity. We are proposing another functionality to the system, that is the social annotation of songs (similar to the *Good Vibrations* plug–in [SACH06]). Based on the manually annotated tags from users, the system would propose a set of tags to the new songs. These tags will be validated by the community. Thus, the system is, semi–automatically, propagating already existing songs' annotations to similar newly crawled songs.

Regarding *Foafing the Music* prototype, there are some tasks to be done. The system is based on a music ontology, but it does not take profit of all the advantages that an ontology can offer. In this sense, more elaborated inferences can be achieved by analyzing user listening habits, and linking it with the semantic metadata of the music titles. It is interesting to analyze, via data mining, existing patterns of listening habits (e.g is there any relationship between the timestamp —when a user listens to a song— and the type of song?). Moreover, collaborative filtering is not yet fully exploited. The main problem is the lack of a large number of users, that makes very difficult to compute a reliable similarity among users. Anyway, we are planning to exploit FOAF profiles to create neighbourhoods. That is, to take into account not only musical taste, but demographic information, geographic data, and general interests. The analysis of the clusters could devise some interesting results about stereotyping and musical tendencies.

8.3 Closing Statement

This Thesis proposes a conceptual framework for a multi–faceted description of music assets. These descriptions are exploited by two implemented prototypes. Most of the problems addressed in these prototypes could be alleviated or would change its focus if music files were enriched with metadata from their own origin (i.e. the recording studio). As this does not seem to be a priority for music technology manufacturers, we foresee a long life to our field, as digital music consumers are asking for the benefits of populating their music collections with a consistent and varied set of semantic descriptors.

Nowadays, we are now viewing an explosion of the practical applications coming out from the MIR research: Music Identification systems, Music Recommenders and Playlist Generators, Music Search Engines, Music Discovery and Personalization systems, and this is just the beginning[2]. At this stage, we might be closer in bridging the semantic gap in music than in any other multimedia knowledge domain. Music was a key factor in taking Internet from its text–centered origins to being a complete multimedia environment. Music might do the same for the Semantic Web.

[2]A detailed list of MIR systems are available at http://mirsystems.info/

List of Figures

List of Tables

Listings

Bibliography

[ABB+03] M. Anderson, M. Ball, H. Boley, S. Greene, N. Howse, D. Lemire, and S. McGrath. Racofi: A rule-applying collaborative filtering system. In *Proceedings of COLA'03*. IEEE/WIC, October 2003.

[ABFS02] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Querying xml sources using an ontology-based mediator. *On the Move to Meaningful Internet Systems 2002, ODBASE02*, pages 429–448, 2002.

[AP02] J.-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In *Proceedings of 3rd International Conference on Music Information Retrieval*, pages 157–163, Paris, France, 2002.

[AP04] J.-J. Aucouturier and F. Pachet. Improving timbre similarity: how high's the sky. In *Journal of Negative Results in Speech and Audio Science*, 2004.

[AvH04] G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, April 2004.

[AZ97] Christopher Avery and Richard Zeckhauser. Recommender systems for evaluating computer messages. *Commun. ACM*, 40(3):88–89, 1997.

[BDDS04] J. P. Bello, C. Duxbury, M. E. Davies, and M. B. Sandler. On the use of phase and energy for musical complex domain. In *IEEE Signal Processing Letters*, pages 533–556, 2004.

[BH03] S. Baumann and O. Hummel. Using cultural metadata for artist recommendations. 2003.

[BH05] S. Baumann and O. Hummel. Enhancing music recommendation algorithms using cultural metadata. *Journal of New Music Research*, 34(2), 2005.

[BL99] Tim Berners-Lee. *Weaving the Web*. Texere Publishing Ltd., November 1999.

[BLEW03] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. In *Proceedings of 4th International Symposium on Music Information Retrieval*, Baltimore, Maryland, 2003.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[BP05] P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of 6th International Conference on Music Information Retrieval*, London, UK, 2005.

[BS03] P. Bello and M. Sandler. Phase-based note onset detection for music signals. In *Proceedings of IEEE ICASSP*, 2003.

[Bur02] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, first edition, 1999.

[CBKH02] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. A review of algorithms for audio fingerprinting. *Workshop on Multimedia Signal Processing*, 2002.

[CCKMB06] P. Cano, O. Celma, M. Koppenberger, and J. Martin-Buldu. Topology of music recommendation networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 16(013107), 2006.

[CDS05] N. Chetry, M. Davies, and M. Sandler. Musical instrument identification using lsf and k-means. In *Proc. of the 118th Convention of the AES*, 2005.

[CF00] W. Cohen and W. Fan. Web-collaborative filtering: recommending music by crawling the web. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):685–698, 2000.

[CGMM99] M. Claypool, A. Gokhale, T. Miranda, and P. Murnikov. Combining content-based and collaborative filters in an online newspape. *Proceedings of ACM SIGIR Workshop on Recommender Systems*, 1999.

[CKH+04] P. Cano, M. Koppenberger, P. Herrera, O. Celma, and V. Tarasov. Sound effect taxonomy management in production environments. In *Proceedings of 25th International AES Conference*, London, UK, 2004.

[CKW+05] P. Cano, M. Koppenberger, N. Wack, J. G. Mahedero, J. Masip, O. Celma, D. Garcia, E. Gomez, F. Gouyon, E. Guaus, P. Herrera, J. Massaguer, B. Ong, M. Ramirez, S. Streich, and X. Serra. An industrial-strength content-based music recommendation system. In *Proceedings of 28th Annual International ACM SIGIR Conference*, Salvador, Brazil, 2005.

[CV00] W. Chai and B. Vercoe. Using user models in music information retrieval systems. *Proceedings of 1st International Conference on Music Information Retrieval*, 2000.

[CXH04] I. Cruz, H. Xiao, and F. Hsu. An ontology-based framework for xml semantic integration. *8th Database Engineering and Applications Symposium*, 2004.

[Dan05] R. Dannenberg. Toward automated holistic beat tracking, music analysis, and understanding. In *Proceedings of 6th International Conference on Music Information Retrieval*, London, UK, 2005.

[dB03] Jos de Bruijn. Using ontologies - enabling knowledge sharing and reuse on the semantic web. Technical Report DERI-2003-10-29, DERI, 2003.

[DE95] D.Maltz and K. Ehrlich. Pointing the way: active collaborative filtering. In *CHI '95: Proceedings of SIGCHI conference on Human factors in computing systems*, pages 202–209, New York, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[DGW04] S. Dixon, F. Gouyon, and G. Widmer. Towards characterization of music via rhythmic patterns. In *Proceedings of 5th International Conference on Music Information Retrieval*, Barcelona, SPAIN, 2004.

[DHL03] M. Doerr, J. Hunter, and C. Lagoze. Towards a core ontology for information integration. *Journal of Digital Information, 4(1)*, 2003.

[DK03] M. Doller and H. Kosch. An mpeg-7 multimedia data cartridge. In *Proceedings of SPIE Conference on Multimedia Computing and Networking 2003 (MMCN03)*, Santa Clara, USA, 2003.

[DP04] M. E. P. Davies and M. D. Plumbley. Causal tempo tracking of audio. In *Proceedings of 5th International Conference on Music Information Retrieval*, Barcelona, SPAIN, 2004.

[EWAS02] D. Ellis, B. Whitman, A.Berenzweig, and S.Lawrence. The quest for ground truth in musical artist similarity. In *Proceedings of 3rd International Symposium on Music Information Retrieval*, pages 170–177, Paris, 2002.

[Foo97] J. Foote. Content-based retrieval of music and audio. *Multimedia Storage and Archiving Systems II. Proceedings of SPIE*, pages 138–147, 1997.

[Gar06] R. Garcia. *A Semantic Web Approach to Digital Rights Management.* PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain., 2006.

[GC05] R. Garcia and O. Celma. Semantic integration and retrieval of multimedia metadata. In *Proceedings of 4rd International Semantic Web Conference. Knowledge Markup and Semantic Annotation Workshop*, Galway, Ireland, 2005.

[GD04] F. Gouyon and S. Dixon. Dance music classification: A tempo-based approach. *Proceedings of 5th International Conference on Music Information Retrieval*, 2004.

[GD05] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29:34–54, 2005.

[GDOT92] D. Goldberg, Nichols D, B. M. Oki, and D. Terry. Collaborative filtering to weave and information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[GH04] E. Gomez and P. Herrera. Estimating the tonality of polyphonic audio files: Cognitive versus machine learning modelling strategies. *Proceedings of 5th International Conference on Music Information Retrieval*, 2004.

[Gom06a] E. Gomez. *Tonal Description of Music Audio Signals.* PhD thesis, 2006.

[Gom06b] E. Gomez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing, Special Cluster on Computation in Music*, 18(3), 2006.

[GP05] J. Golbeck and B. Parsia. Trust network-based filtering of aggregated claims. In *International Journal of Metadata, Semantics, and Ontologies*, 2005.

[Gru93] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[Her06] P. Herrera. *Automatic classification of percussion sounds: from acoustic features to semantic descriptions*. PhD thesis, 2006.

[HIMT03] A.Y. Halevy, Z.G. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. *12th World Wide Web Conference*, pages 556–567, 2003.

[HKD06] P. Herrera, A. Klapuri, and M. Davy. Automatic classification of pitched musical instrument sounds. *Signal processing methods for music transcription, Springer*, 29:34–54, 2006.

[HKTR04] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[HL02] G. Hu and Q. Li. Schema validation applied to native xml databases. In *International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet*, L'Aquila, Italy, 2002.

[HMI03] K. Hoashi, K. Matsumoto, and N. Inoue. Personalization of user profiles for content-based music retrieval based on relevance feedback. In *MULTIMEDIA '03: Proceedings of eleventh ACM international conference on Multimedia*, pages 110–119, New York, NY, USA, 2003. ACM Press.

[HS05] C. A. Harte and M. Sandler. Automatic chord identification using a quantised chromagram. *Proc. of the 118th Convention. of the AES*, 2005.

[HSG04] P. Herrera, V. Sandvold, and F. Gouyon. Percussion-related semantic descriptors of music audio files. In *Proceedings of 25th International AES Conference*, London, UK, 2004.

[HSRF95] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of SIGCHI conference on Human factors in computing systems*, pages 194–201, New York, USA, 1995.

[Hun99] J. Hunter. A proposal for an mpeg-7 description definition language. *MPEG-7 AHG Test and Evaluation Meeting*, 1999.

[Hun01] J. Hunter. Adding multimedia to the semantic web - building an mpeg-7 ontology. *International Semantic Web Working Symposium*, 2001.

[Jac04] M. Jacob. Managing large sound databases using mpeg-7. In *Proceedings AES 25th Internationl Conference*, London, UK, 2004.

[Kle02] M.C.A. Klein. Interpreting xml documents via an rdf schema ontology. *ECAI Workshop on Semantic Authoring, Annotation & Knowledge Markup*, 2002.

[Kno04] I. Knopke. Arooooga: An audio search engine for the world wide web. In *Proceedings of 5th International Conference on Music Information Retrieval*, Barcelona, Spain, 2004.

[Kos04] H. Kosch. *Distributed Multimedia Database Systems supported by MPEG-7 and MPEG-21.* CRC Press, 2004.

[LM05] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[LS01] B. Logan and A. Salomon. A music similarity function based on signal analysis. *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001*, pages 745–748, 2001.

[LS03] L. Lakshmanan and F. Sadri. Interoperability on xml data. *International Semantic Web Symposium, ISWC*, pages 146–163, 2003.

[LVH05] T. W. Leong, F. Vetere, and S. Howard. The serendipity shuffle. In *OZCHI '05: Proceedings of 19th conference of the computer-human interaction special interest group*, pages 1–4, Narrabundah, Australia, Australia, 2005.

[Mil95] G. A. Miller. WordNet: A lexical database for english. *Communications of the ACM*, pages 39–45, November 1995.

[MLdlR03] M. Montaner, B. Lopez, and J. Lluis de la Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, (19):285–330, 2003.

[MS02] Salembier P. Manjunath, B. S. and T. Sikora. *Introduction to MPEG 7: Multimedia Content Description Language.* Ed. Wiley, 2002.

[OH05] B. Ong and P. Herrera. Semantic segmentation of music audio contents. *Proceeding of International Computer Music Conference*, 2005.

[ONH03] J. v. Ossenbruggen, F. Nack, and L. Hardman. That obscure object of desire: Multimedia metadata on the web (part i & ii). *Information Systems Report INS-E0308*, 2003.

[Pac05] F. Pachet. *Knowledge Management and Musical Metadata.* Idea Group, 2005.

[Pam04] E. Pampalk. A matlab toolbox to compute music similarity from audio. 2004.

[Pam06] E. Pampalk. *Computational Models of Music Similarity and their Application to Music Information Retrieval.* PhD thesis, 2006.

[Paz99] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. In *Artificial Intelligence Review*, volume Vol. 13, Numbers 5-6, pages 393–408, 1999.

[PBM+02] J. Pickens, J. P. Bello, G. Monti, T. Crawford, M. Dovey, M. Sandler, and D. Byrd. Polyphonic score retrieval using polyphonic audio queries: A harmonic modelling approach. *Proceedings of 3rd International Conference on Music Information Retrieval*, pages 140–149, 2002.

[PdRME04] E. Perik, B. de Ruyter, P. Markopoulos, and B. Eggen. The sensitivities of user profile information in music recommender systems. In *Proceedings of Private, Security, Trust*, 2004.

[Por80] M. F. Porter. An algorithm for suffix stripping. In *Program 14*, pages 130–137, 1980.

[PSP02] Patel-Schneider and J. P.F., Simeon. The yin/yang web: Xml syntax and rdf semantics. *11th World Wide Web Conference*, 2002.

[PWL01] F. Pachet, G. Westermann, and D. Laigre. Musical data mining for electronic music distribution. 2001.

[Ric79] E. Rich. User modeling via stereotypes. In *Cognitive Science: A Multidisciplinary Journal*, volume Vol. 3, No. 4, pages 329–354, 1979.

[RIS+94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186. ACM, ACM Press, 1994.

[RV97] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[SACH06] V. Sandvold, T. Aussenac, O. Celma, and P. Herrera. Good vibrations: Music discovery through personal musical concepts. In *Proceedings of 7th International Conference on Music Information Retrieval (to be published)*, Victoria, Canada, 2006.

[SH04] V. Sandvold and P. Herrera. Towards a semantic descriptor of subjective intensity in music. In *Proceedings of 5th International Conference on Music Information Retrieval*, Barcelona, 2004.

[Sha94] U. Shardanand. Social information filtering for music recommendation. Master's thesis, Massachussets Institute of Technology, September 1994.

[SKKR01] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW'01: Proceedings of 10th International Conference on World Wide Web*, pages 285–295, 2001.

[SKW05a] M. Schedl, P. Knees, and G. Widmer. A web-based approach to assessing artist similarity using co-occurrences. In *Proceedings of 4th International Workshop on Content-Based Multimedia Indexing (CBMI'05)*, June 2005.

[SKW05b] Markus Schedl, Peter Knees, and Gerhard Widmer. Improving prototypical artist detection by penalizing exorbitant popularity. In *Proceedings of 3rd International Symposium on Computer Music Modeling and Retrieval*, pages 196–200, 2005.

[SLH06] N. Shadbolt, Tim B. Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[SM86] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[SM95] U. Shardanand and P. Maes. Social information filtering: Algorithms for automaing "word of mouth". In *Proceedings of CHI'95*, 1995.

[SS01] K. Swearingen and R. Sinha. Beyond algorithms: An hci perspective on recommender systems. In *ACM SIGIR. Workshop on Recommender Systems*, volume Vol. 13, Numbers 5-6, pages 393–408, 2001.

[STH+99] J. Shanmugasundaram, K. Tufte, G. He, DeWitt D. Zhang, C., and J. Naughton. Relational databases for querying xml documents: Limitations and opportunities. In *Proceedings of 25th International Conference on Very Large Data Bases*, Edinburgh, Scotland, 1999.

[SWS+00] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[TDCZ02] F. Tian, D. DeWitt, J. Chen, and C. Zhang. *The Design and Performance Evaluation of Alternative XML Storage Strategies*, volume 31, Number 1. ACM Sigmod Record, 2002.

[TPS04] C. Tsinaraki, P. Polydoros, and Christodoulakis S. Integration of owl ontologies in mpeg-7 and tvanytime compliant semantic indexing. *Advanced Information Systems Engineering, 16th International Conference, CAiSE04*, pages 398–413, 2004.

[Tro03] R. Troncy. Integrating structure and semantics into audio-visual documents. *Proceedings of 2nd International Semantic Web Conference*, 2003.

[TS05] C. Tsinaraki and Christodoulakis S. Semantic user preference descritpions in mpeg-7/21. 2005.

[Tza02] George Tzanetakis. *Manipulation, analysis and retrieval systems for audio signals*. PhD thesis, 2002. Adviser-Perry Cook.

[UvS02] A. Uitdenbogerd and R. van Schnydel. A review of factors affecting music recommender success. In *Proceedings of 3rd International Conference on Music Information Retrieval*, Paris, France, 2002.

[VB04] S. Vembu and S. Baumann. A self-organizing map based knowledge discovery for music recommendation systems. In *Computer Music Modeling and Retrieval*, Esbjerg, Denmark, 2004.

[WK96] G. Webb and M. Kuzmycz. Feature based modelling: A methodology for producing coherent, consistent, dynamically changing models of agents' competencies. In *User Modeling and User-Adapted Interaction*, pages 117–150, 1996.

[WK03] U. Westermann and W. Klas. *An analysis of XML database solutions for the management of MPEG-7 media descriptions*, volume 35, Issue 4. ACM Computing Surveys (CSUR), 2003.

[WL02] B. Whitman and S. Lawrence. Inferring descriptions and similarity for music from community metadata. In *Proceedings of International Computer Music Conference*, Goteborg, Sweden, 2002.

[WPVS05] G. Wijnalda, S. Pauws, F. Vignoli, and H. Stuckenschmidt. A personalized music system for motivation in sport performance. *IEEE Pervasive Computing*, 4(3):26–32, 2005.

[YGO04] K. Yoshii, M. Goto, and H. G. Okuno. Automatic drum sound description for real-world music using template adaptation and matching methods. *Proceedings of 5th International Conference on Music Information Retrieval*, 2004.

[ZF04] M. Zadel and I. Fujinaga. Web services for music information retrieval. In *Proceedings of 5th International Conference on Music Information Retrieval*, Barcelona, 2004.

[ZP03] A. Zils and F. Pachet. Extracting automatically the perceived intensity of music titles. *Proceedings of DAFX-03*, 2003.

A Music Ontology described in OWL DL

This appendix contains the OWL–DL ontology used by the *Foafing the Music* application (see chapter 7).

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE owl [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#"> ]>
<rdf:RDF
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://foafing-the-music.iua.upf.edu/music-ontology#"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  xmlns:wordnet="http://xmlns.com/wordnet/1.6/"
  xmlns:mm="http://musicbrainz.org/mm/mm-2.1#"
>

  <owl:Ontology rdf:about="">
    <owl:versionInfo>v0.2 2006/05/17 ocelma</owl:versionInfo>
    <rdfs:comment>Foafing the Music Ontology (Version 0.2)</
       rdfs:comment>
  </owl:Ontology>

<!--          -->
<!-- CLASSES -->
<!--          -->
  <!-- Artist -->
  <owl:Class rdf:ID="Artist">
    <rdfs:subClassOf rdf:resource="http://xmlns.com/wordnet/1.6/
       Artist"/>
    <owl:equivalentClass rdf:resource="http://musicbrainz.org/mm/
       mm-2.1#Artist"/>
  </owl:Class>

  <!-- Track -->
  <owl:Class rdf:ID="Track">
    <rdfs:subClassOf rdf:resource="http://xmlns.com/wordnet/1.6/
       Song"/>
```

```
    <owl:equivalentClass rdf:resource="http://musicbrainz.org/mm/
       mm-2.1#Track"/>
  </owl:Class>

<!--                    -->
<!-- Data Properties    -->
<!--                    -->

<!--         -->
<!-- Artist  -->
<!--         -->
  <owl:DatatypeProperty rdf:ID="name">
    <rdfs:domain rdf:resource="#Artist"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>name</rdfs:label>
  </owl:DatatypeProperty>

  <!-- decades -->
  <owl:DatatypeProperty rdf:ID="decades">
    <rdfs:domain rdf:resource="#Artist"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdfs:label>decades</rdfs:label>
  </owl:DatatypeProperty>

  <!-- genre -->
  <owl:DatatypeProperty rdf:ID="genre">
    <rdfs:domain rdf:resource="#Artist"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdfs:label>genre</rdfs:label>
  </owl:DatatypeProperty>

  <!-- city -->
  <owl:DatatypeProperty rdf:ID="city">
    <rdfs:domain rdf:resource="#Artist"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdfs:label>city</rdfs:label>
  </owl:DatatypeProperty>

  <!-- nationality -->
  <owl:DatatypeProperty rdf:ID="nationality">
    <rdfs:domain rdf:resource="#Artist"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdfs:label>nationality</rdfs:label>
  </owl:DatatypeProperty>

<!--        -->
<!-- Track -->
```

```
<!--       -->
  <!-- title -->
  <owl:DatatypeProperty rdf:ID="title">
    <rdfs:domain rdf:resource="#Track"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>title</rdfs:label>
  </owl:DatatypeProperty>

  <!-- Artist plays a Track-->
  <owl:ObjectProperty rdf:ID="plays">
    <rdfs:domain rdf:resource="#Artist"/>
    <rdfs:range rdf:resource="#Track"/>
  </owl:ObjectProperty>

  <!-- Track played by an Artist -->
  <owl:ObjectProperty rdf:ID="playedBy">
    <rdfs:domain rdf:resource="#Track"/>
    <rdfs:range rdf:resource="#Artist"/>
    <owl:inverseOf rdf:resource="#plays" />
  </owl:ObjectProperty>

  <!-- fingerprint -->
  <owl:DatatypeProperty rdf:ID="fingerprint">
    <rdfs:domain rdf:resource="#Track"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>fingerprint</rdfs:label>
  </owl:DatatypeProperty>

  <!-- itms -->
  <owl:DatatypeProperty rdf:ID="itms">
    <rdfs:domain rdf:resource="#Track"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>itms</rdfs:label>
    <rdfs:comment>iTunes Identifier</rdfs:comment>
  </owl:DatatypeProperty>

  <!-- duration -->
  <owl:DatatypeProperty rdf:ID="duration">
    <rdfs:domain rdf:resource="#Track"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       float"/>
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>duration</rdfs:label>
  </owl:DatatypeProperty>

  <!-- Descriptor: Base Data Property -->
  <owl:DatatypeProperty rdf:ID="descriptor">
```

```
    <rdfs:domain rdf:resource="#Track" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
       string"/>
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>Basic descriptor</rdfs:label>
  </owl:DatatypeProperty>

  <!-- Tonality Descriptors -->
  <owl:DatatypeProperty rdf:ID="tonalityDescriptor" >
    <rdfs:subPropertyOf rdf:resource="#descriptor" />
  </owl:DatatypeProperty>
    <!-- Tonalness -->
    <owl:DatatypeProperty rdf:ID="tonalness" >
      <rdfs:subPropertyOf rdf:resource="#tonalityDescriptor" />
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
         float"/>
      <rdfs:label>tonalness</rdfs:label>
      <rdfs:comment>This descriptor tells how tonal is the song</
         rdfs:comment>
    </owl:DatatypeProperty>
    <!-- Mode -->
    <owl:DatatypeProperty rdf:ID="keyMode">
      <rdfs:subPropertyOf rdf:resource="#tonalityDescriptor" />
      <rdfs:range>
        <owl:DataRange>
          <owl:oneOf rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/
               XMLSchema#string">Minor</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:first rdf:datatype="http://www.w3.org/2001/
                 XMLSchema#string">Major</rdf:first>
              <rdf:rest rdf:resource="http://www.w3.org
                 /1999/02/22-rdf-syntax-ns#nil"/>
            </rdf:rest>
          </owl:oneOf>
        </owl:DataRange>
      </rdfs:range>
      <rdfs:label>Key Mode</rdfs:label>
    </owl:DatatypeProperty>
    <!-- Key -->
    <owl:DatatypeProperty rdf:ID="key">
      <rdfs:subPropertyOf rdf:resource="#tonalityDescriptor" />
      <rdfs:label>key</rdfs:label>
      <rdfs:range>
        <owl:DataRange>
          <owl:oneOf rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/
               XMLSchema#string">A</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:first rdf:datatype="http://www.w3.org/2001/
                 XMLSchema#string">A#/Bb</rdf:first>
              <rdf:rest rdf:parseType="Resource">
```

```
                <rdf:first rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string">B</rdf:first>
                <rdf:rest rdf:parseType="Resource">
                  <rdf:first rdf:datatype="http://www.w3.org
                      /2001/XMLSchema#string">C</rdf:first>
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:first rdf:datatype="http://www.w3.org
                       /2001/XMLSchema#string">C#/Db</rdf:first>
                    <rdf:rest rdf:parseType="Resource">
                      <rdf:first rdf:datatype="http://www.w3.org
                         /2001/XMLSchema#string">D</rdf:first>
                      <rdf:rest rdf:parseType="Resource">
                        <rdf:first rdf:datatype="http://www.w3.
                           org/2001/XMLSchema#string">D#/Eb</
                           rdf:first>
                        <rdf:rest rdf:parseType="Resource">
                          <rdf:first rdf:datatype="http://www.w3.
                             org/2001/XMLSchema#string">F</
                             rdf:first>
                          <rdf:rest rdf:parseType="Resource">
                            <rdf:first rdf:datatype="http://www.
                               w3.org/2001/XMLSchema#string">E</
                               rdf:first>
                            <rdf:rest rdf:parseType="Resource">
                              <rdf:first rdf:datatype="http://www
                                 .w3.org/2001/XMLSchema#string">
                                 F#/Gb</rdf:first>
                              <rdf:rest rdf:parseType="Resource">
                                <rdf:first rdf:datatype="http://
                                   www.w3.org/2001/XMLSchema#
                                   string">G#/Ab</rdf:first>
                                <rdf:rest rdf:resource="http://
                                   www.w3.org/1999/02/22-rdf-
                                   syntax-ns#nil"/>
                              </rdf:rest>
                            </rdf:rest>
                          </rdf:rest>
                        </rdf:rest>
                      </rdf:rest>
                    </rdf:rest>
                  </rdf:rest>
                </rdf:rest>
              </rdf:rest>
            </owl:oneOf>
          </owl:DataRange>
        </rdfs:range>
      </owl:DatatypeProperty>

      <!-- Rhythm Descriptors -->
      <owl:DatatypeProperty rdf:ID="rhythmDescriptor" >
        <rdfs:subPropertyOf rdf:resource="#descriptor" />
      </owl:DatatypeProperty>
```

```
    <!-- swing Ratio -->
    <owl:DatatypeProperty rdf:ID="swingRatio">
      <rdfs:subPropertyOf rdf:resource="#rhythmDescriptor" />
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
          int"/>
      <rdfs:label>Swing Ratio</rdfs:label>
    </owl:DatatypeProperty>
    <!-- Tempo -->
    <owl:DatatypeProperty rdf:ID="tempo">
      <rdfs:subPropertyOf rdf:resource="#rhythmDescriptor" />
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
          int"/>
      <rdfs:label>Tempo</rdfs:label>
    </owl:DatatypeProperty>
    <!-- Meter -->
    <owl:DatatypeProperty rdf:ID="meter">
      <rdfs:subPropertyOf rdf:resource="#rhythmDescriptor" />
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
          int"/>
      <rdfs:label>Meter</rdfs:label>
    </owl:DatatypeProperty>

  <!-- Dynamic Complexity -->
  <owl:DatatypeProperty rdf:ID="dynamicComplexity">
    <rdfs:subPropertyOf rdf:resource="#descriptor" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
        float"/>
    <rdfs:label>Dynamic Complexity</rdfs:label>
  </owl:DatatypeProperty>

  <owl:DatatypeProperty rdf:ID="Undanceability" >
    <rdfs:subPropertyOf rdf:resource="#dynamicComplexity" />
    <rdfs:subPropertyOf rdf:resource="#rhythmDescriptor" />
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
      decimal" />
  </owl:DatatypeProperty>

<!--                    -->
<!-- Object Properties -->
<!--                    -->
  <!-- Artist relationships -->
  <owl:ObjectProperty rdf:ID="relatedWith">
    <rdfs:domain rdf:resource="#Artist"/>
    <rdfs:range rdf:resource="#Artist"/>
    <rdfs:label>related with</rdfs:label>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="soundsLike">
   <rdfs:domain rdf:resource="#Artist"/>
   <rdfs:range rdf:resource="#Artist"/>
   <rdfs:label>sounds like</rdfs:label>
  </owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="followersOf">
 <rdfs:domain rdf:resource="#Artist"/>
 <rdfs:range rdf:resource="#Artist"/>
 <rdfs:label>followers of</rdfs:label>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasInfluenced">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource="#Artist"/>
  <rdfs:label>has influenced</rdfs:label>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="workedWith">
  <rdfs:domain rdf:resource="#Artist"/>
  <rdfs:range rdf:resource="#Artist"/>
  <rdfs:label>worked with </rdfs:label>
</owl:ObjectProperty>
```

Related publications by the author

In this annex, we provide a list of publications which are relevant to this Thesis, and the author has participated. Electronic versions of most of these publications, as well as a list of other publications from the author non related to this dissertation are available from `http://mtg.upf.edu`.

- **Celma, O**. Ramirez, M. Herrera, P. 2005. "Getting music recommendations and filtering newsfeeds from FOAF descriptions". Proceedings of Workshop on Scripting for the Semantic Web; Heraklion, Greece.

  Abstract:

  This document proposes to use the Friend of a Friend (FOAF) definition to recommend music depending on user's musical tastes and to filter music-related newsfeeds. One of the goals of the project is to explore music content discovery, based on both user profiling -FOAF descriptions- and content-based descriptions -extracted from the audio itself.

  Related to chapters 3 and 7.

- Herrera, P. Bello, J. Widmer, G. Sandler, M. **Celma, O**. Vignoli, F. Pampalk, E. Cano, P. Pauws, S. Serra, X. 2005. "SIMAC: Semantic interaction with music audio contents". Proceedings of 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies; Savoy Place, London, UK.

  Abstract:

  The SIMAC project addresses the study and development of innovative components for a music information retrieval system. The key feature is the usage and exploitation of semantic descriptors of musical content

that are automatically extracted from music audio files. These descriptors are generated in two ways: as derivations and combinations of lower-level descriptors and as generalizations induced from manually annotated databases by the intensive application of machine learning. The project aims also towards the empowering (i.e. adding value, improving effectiveness) of music consumption behaviours, especially of those that are guided by the concept of similarity.

Related to chapter 4.

- Cano, P. Koppenberger, M. Wack, N. G. Mahedero, J. Masip, J. **Celma, O**. Garcia, D. Gomez, E. Gouyon, F. Guaus, E. Herrera, P. Massaguer, J. Ong, B. Ramirez, M. Streich, S. Serra, X. 2005. "An Industrial-Strength Content-based Music Recommendation System". Proceedings of 28th Annual International ACM SIGIR Conference; Salvador, Brazil.

  Abstract:

  We present a metadata free system for the interaction with massive collections of music, the *MusicSurfer*. *MusicSurfer* automatically extracts descriptions related to instrumentation, rhythm and harmony from music audio signals. Together with efficient similarity metrics, the descriptions allow navigation of multimillion track music collections in a flexible and efficient way without the need of metadata or human ratings.

  Related to chapter 4.

- Garcia, R. **Celma, O**. 2005. "Semantic Integration and Retrieval of Multimedia Metadata". Proceedings of 4rd International Semantic Web Conference; Galway, Ireland.

  Abstract:

  The amount of digital media that has to be actually managed has already become unaffordable without fine-grained computerised support. This requires an extensive use of multimedia metadata. MPEG-7 is the greatest metadata framework created to date but it is based on XML Schemas. Therefore, its does not have formal semantics, which makes difficult to manage, extend and integrate it. Consequently, there have been a lot attempts to move MPEG-7 to the Semantic Web. Our approach contributes a complete and automatic mapping of the whole MPEG-7 standard to

OWL. It is based on a generic XML Schema to OWL mapping. The previous mapping is complemented with an XML metadata instances to RDF mapping that completes a transparent transfer of metadata from the XML to the Semantic Web domain. Once in a semantic space, data integration, which is a crucial factor when several sources of information are available, is facilitated enormously. We have used the generated MPEG-7 OWL ontology as an "upper-ontology" for multimedia metadata, where three different music schemas have been linked. Thus, it has been possible to retrieve related information from instances of all the metadata sources. Furthermore, detecting and merging instances from different sources has allowed us to enhance the description of audio files, both content-based and editorial data.

Related to chapter 5.

- **Celma, O**. Herrera, P. Serra, X. 2006. "Bridging the Music Semantic Gap". The first International Conference on Semantics and Digital Media Technology; Athens, Greece.

  Abstract:

  In this paper we present the music information plane and the different levels of information extraction that exist in the musical domain. Based on this approach we propose a way to overcome the existing semantic gap in the music field. Our approximation is twofold: we propose a set of music descriptors that can automatically be extracted from the audio signals, and a top-down approach that adds explicit and formal semantics to these annotations. These music descriptors are generated in two ways: as derivations and combinations of lower-level descriptors and as generalizations induced from manually annotated databases by the intensive application of machine learning algorithms. We believe that merging both approaches (bottom-up and top-down) can overcome the existing semantic gap in the musical domain.

  Related to chapter 5.

- **Celma, O**. Mieza, E. 2004. "An Opera Information System Based on MPEG-7". Proceedings of 25th International AES Conference; London, UK.

  Abstract:

We present an implementation of the MPEG-7 standard for a multimedia content description of lyric opera in the context of the European IST project: OpenDrama. The project goals are the definition, development, and integration of a novel platform to author and deliver the rich cross-media digital objects of lyric opera. MPEG-7 has been used in OpenDrama as the base technology for a music information retrieval system. In addition to the MPEG-7 multimedia description scheme, different classification schemes have been proposed to deal with operatic concepts such as musical forms (acts, scenes, frames, introduction, etc.), musical indications (piano, forte, ritardando, etc.), and genre and creator roles (singers, musicians, production staff, etc.). Moreover, this project has covered the development of an authoring tool for an MPEG-7 standard, namely MDTools, which includes segmentation, classification scheme generation, creation and production, and media information descriptors.

Related to chapter 5.

- Wust, O. **Celma, O**. 2004. "An MPEG-7 Database System and Application for Content-Based Management and Retrieval of Music". Proceedings of Fifth International Conference on Music Information Retrieval; Barcelona.

  Abstract:

  Computer users are gaining access to and are starting to accumulate moderately large collections of multimedia files, in particular of audio content, and therefore demand new applications and systems capable of effectively retrieving and manipulating these multimedia objects. Content-based retrieval of multimedia files is typically based on searching within a feature space, defined as a collection of parameters that have been extracted from the content and which describe it in a relevant way for that particular retrieval application. The MPEG-7 standard offers tools to model these metadata in an interoperable and extensible way, and can therefore be considered as a framework for building content-based audio retrieval systems. This paper highlights the most relevant aspects considered during the design and implementation of a DBMS-driven MPEG-7 layer on top of which a content-based music retrieval system has been built. A particular focus is set on the data modeling and database architechture issues.

Related to chapter 5.

- **Celma, O**. 2004. "Architecture for an MPEG-7 Digital Library". Proceedings of Fifth International Conference on Music Information Retrieval; Barcelona.

  Abstract:

  The MPEG-7 standard provides description mechanisms and taxonomy management for multimedia documents. There are several approaches to design a multimedia database system using MPEG-7 descriptors. We discuss two of them: relational databases and native XML databases. We have implemented an search and retrieval web system for MPEG-7 descriptions based on the latter.

  Related to chapter 5.

- **Celma, O**. Cano, P. Herrera, P. 2006. "Search Sounds: An audio crawler focused on weblogs". Proceedings of 7th Intl. Conference on Music Information Retrieval; Victoria, Canada.

  Abstract:

  In this paper we present a focused audio crawler that mines audio weblogs (MP3 blogs). This source of semi–structured information contains links to audio files, plus some textual information that is referring to the media file. A retrieval system —that exploits the mined data— fetches relevant audio files related to user's text query. Based on these results, the user can navigate and discover new music by means of content-based audio similarity.

  Related to chapter 6.

- **Celma, O**. Ramirez, M. Herrera, P. 2005. "Foafing the music: A music recommendation system based on RSS feeds and user preferences". Proceedings of 6th International Conference on Music Information Retrieval; London, UK.

  Abstract:

  In this paper we give an overview of the Foafing the Music system. The system uses the Friend of a Friend (FOAF) and Rich Site Summary (RSS) vocabularies for recommending music to a user, depending on her musical tastes. Music information (new album releases, related artists'news

and available audio) is gathered from thousands of RSS feeds —an XML format for syndicating Web content. On the other hand, FOAF documents are used to define user preferences. The presented system provides music discovery by means of: user profiling -defined in the user's FOAF description-, context-based information -extracted from music related RSS feeds- and content-based descriptions -extracted from the audio itself.

Related to chapter 7.

- Cano, P. **Celma, O**. Koppenberger, M. Martin-Buldu, J. 2006. "Topology of music recommendation networks". Chaos: An Interdisciplinary Journal of Nonlinear Science Vol.16 .013107.

  Abstract:

  We study the topology of several music recommendation networks, which arise from relationships between artist, co-occurrence of songs in play lists or experts' recommendation. The analysis uncovers the emergence of complex network phenomena in these kinds of recommendation networks, built considering artists as nodes and their resemblance as links. We observe structural properties that provide some hints on navigation and possible optimizations on the design of music recommendation systems. Finally, the analysis derived from existing music knowledge sources provides a deeper understanding of the human music similarity perception.

  Related to chapter 8.