Hybrid Content-Based Collaborative-Filtering Music Recommendations

Master Thesis

Author:	Hugo Siles Del Castillo
Date:	November 6, 2007
University:	University of Twente, The Netherlands
Department:	Computer Science Information System Engineering (ISE)

Graduation Committee:	Prof.dr. Peter M.G. Apers	University of Twente
	Dr.ir. Djoerd Hiemstra	University of Twente
	Riham Adbel Kader, MSc.	University of Twente

Acknowledgements

The present work is the final result of my Master thesis project for the program of Computer Science, Information Systems Engineering track in the University of Twente, The Netherlands.
I would like to thank to all my friends and people who help me in one way or another during my two year's stay in Enschede. Without their friendship and support it would have been really hard to complete successfully this program. Besides, some special thanks to Dan Ellis, professor at the Columbia University in the city of New York, USA, for helping me with the recollection of the audio features from the USPOP collection.
Finally, but perhaps the most important one, I would really like to thank my family for all the support during this two years, especially to my father and my sister for giving me the chance to finish my mater program successfully. Without them this project would have been simply impossible.

Thank you.

Abstract

Recommendation of music is emerging with force nowadays due to the huge amount of music content and because users normally do not have the time to search through these collections looking for new items.

The main purpose of a recommendation system is to estimate the user's preferences and present him with some items that he does not know yet. Currently, most of the audio recommendation systems can be classified in two major kinds: recommendation systems based on collaborative filtering techniques and content based recommendation system. While both kinds of systems have good characteristics, they fail to provide good recommendation is specific situations. Recently a new kink of recommendation systems is emerging, hybrid content-based collaborative-filtering recommendation systems.

The main objective of this thesis is to try to probe that the main disadvantages of content-based and collaborative filtering recommendation methods can be solver using hybrid methods. In order to do this we built a hybrid content-based collaborative filtering recommendation system. This hybrid system possesses the best characteristics of both methods and produce better results than each method individually. To achieve our goal we will also present a research on current content-based and collaborative methods in such a way that we will be able to fin out advantages and disadvantages of them

Contents

1 Introduction

In the last years the widespread of several technologies have changed the way people manage, access, and distribute multimedia content. Technologies such as the development and dissemination of P2P networks, and the increase in storage capacity of portable devices had special effect in the worldwide diffusion of multimedia content. Among all the many kinds of multimedia content, music is one of the most popular content nowadays [1]. The reason for this is that music is an art and can be shared by many people from different countries, languages, and cultural backgrounds. One point of reference for this affirmation is the number of items sold daily by web-based dealers, or the number of items downloaded and shared via the internet. In its 2007 Digital Music Report, the International Federation of the Phonographic Industry (IFPI), stated that the number of legally downloaded songs in 2006 reach the amount of 795 millions. These facts show us that music is commercially and culturally important [7].

As the amount of audio content available is increasing several questions arise on how to efficiently access, discover, and present it to the final user [8]. In order to answer all these questions there is the need for new techniques for classifying, searching and retrieving, and recommending audio content. In this thesis we will focus mainly on techniques for recommending audio contents to users. Commercial applications such as content-based music recommendation systems may become increasingly important component of e-commerce applications. One of the advantages of these applications is that they do not need a lot of effort from the user, who is simply presented with potentially relevant items [7].

Recommendation of music is emerging with force nowadays due to the huge amount of music content and because users normally don't have the time to search through these collections looking for new items. The main purpose of a recommendation system is to estimate the user's preferences and present him with some items that he doesn't know yet. Currently, most of the audio recommendation systems can be classified in two major kinds. Recommendation systems based on collaborative filtering techniques and content-based recommendation systems [9]. While both kinds of systems have good characteristics, they fail to provide good recommendations in specific situations. Recently a new kind of recommendation systems is emerging, hybrid content-based collaborative filtering recommendation systems. In the next paragraphs we will explain each of these kinds briefly.

The main idea of collaborative filtering (CF) methods is to recommend items to a user by taking into account the rates on those items made by other users. CF systems work by collecting user feedback in the form of ratings and exploit similarities and differences among profiles of several users [10]. According to [12] this feedback can be given explicitly, ratings or annotation, or implicitly such as the time spent in examining the content of the recommendations. Although CF has proved to provide good recommendation it presents some troubles. One of the major disadvantages of collaborative filtering is that it cannot recommend new items since the new items don't have any rate. Another disadvantage of CF is that users need to be involved a lot with the system and provide a lot of rates. This is a disadvantage because it is hard to obtain reliable information from many users.

On the other hand, content-based methods provide recommendations by comparing representations of content contained in an item to representations of content rated by the user [10]. In order to obtain a representation of the music content, it is necessary to automatically extract features from the audio signals. These features should be as general as possible and try to represent semantically meaningful concepts. To compare the representations of music content it is required to develop efficient similarity metrics. Recently much effort has been put in computational modeling of music similarity. The area of Music Information Retrieval (MIR) is an emerging, interdisciplinary research field that deals with the way of efficiently representing and finding similarities among music. Applications in this field range from automated music analysis to personalized music recommendation, online music access, query-based retrieval, and automatic playlist generation [14], [15]. A good overview of current MIR tasks can be found by looking at the MIREX competition for MIR algorithms at [16].

In general current content-based systems have not yet had significant impact on society due to an inability to bridge the semantic gap between computers and humans [2]. It is hard to extract features from the audio signals that have a truly human meaning. Furthermore, the work presented in [37] states that users put a great value in online music reviews, ratings, recommendations, etc. It seems that the information needs of people are quite related with social behavior and not only based on content or features of music.

From the above paragraphs we can see that both collaborative filtering and content-based recommendation systems present problems that are hard to solve. One possible way to overcome these problems is the use of hybrid methods that connect collaborative filtering and content-based methods. The main idea of hybrid recommendations is to reflect both ratings and content data in modeling the user's preferences. Nevertheless, one problem is that representations of user preferences are different in both methods. Content-based methods represent the preferences as a set of features while collaborative filtering represents the preferences as a set of ratings [9]. In order to mix both methods we have to deal with ad-hoc rules to joint these two kinds of representations.

This thesis will try to solve the main disadvantages of content-based and collaborative filtering recommendation methods. With the purpose of overcoming these disadvantages a hybrid content-based collaborative filtering recommendation system will be built. This hybrid system should possess the best characteristics of both methods and will produce better results than each method individually. To achieve these goals we will also present research on current content-based and collaborative methods in such a way that we will be able to find out advantages and disadvantages of them.

1.1 Research Question

The main research question of this thesis will be stated as follows:

> **Is it possible to obtain better results in music recommendation with hybrid methods rather than with pure content-based or with pure collaborative filtering methods?**

In order to answer this question we will have to narrow down the field of research. For this purpose we pose some specific research sub-questions that will guide the process of research. These sub-questions will be answered in the final conclusions of this thesis and they are:

a) What are the advantages and disadvantages of using pure collaborative filtering or pure content-based methods in music recommendation?

b) What are the main advantages and disadvantages of using hybrid systems for recommending music?

c) What is the performance of pure collaborative filtering or pure content-based methods in music recommendation?

d) What is the performance of hybrid systems in music recommendation and to what extent do they produce better results than pure methods?

1.2 Research Approach

In order to answer our main research question, we will start this project by answering the posed sub-question. Once we have a proper answer for all the sub-questions we will start the design and development of the system.

a) The first question will be answered by carrying on a study of literature. This study will give a good overview of pure collaborative filtering and content-based recommendation methods.

b) The second sub-question will be answered with a literature study of current hybrid systems. Besides, the development of a hybrid recommendation system will let us probe the theoretical advantages or disadvantages.

c) In order to measure the performance of pure recommendation methods we will develop small prototypes of these systems and perform a series of experiments. With these experiments we will obtain enough data to compare these methods.

d) The last sub-question of this project will be answered by using the developed system to perform several experiments in order to measure the performance of hybrid recommendation systems. Once we obtain a measure of the performance, we will be able to compare these results with the results obtained from pure recommendation methods.

1.3 Intended audience

The work presented in this thesis is intended for persons with some interest in the area of music information retrieval, students from college or master programs in computer science, etc. Some basic knowledge about information retrieval may be required. Besides, knowledge on signal processing and machine learning techniques is desirable.

1.4 Structure of this thesis

This document will be structured in the following way. Chapter 2 and 3 will cover some related work and theories behind pure content-based, pure collaborative filtering, and hybrid recommendation systems. In chapter 4 we will discuss different methods to perform evaluation of recommendation systems and we will explain the kind of evaluations. In chapter 5 an explanation of the different prototypes and experiments done during the development of the thesis will be given. Chapter 6 will present the results of the experiments and explain in detail their meaning. Finally, in chapter 7 some conclusions will be drawn and we will talk about some possible future work in this area.

2 State of the art

2.1 Information Overload and Information Retrieval

As explained in chapter 1, the increase and widespread of different technologies have changed the way people access, manages, and distributes multimedia content. One direct consequence of this is the corresponding information overload that affects any user when looking for some specific or new items. The work presented in [34] defines information overload as:

> "…*the state of having too much information to make a decision or remain informed about a topic. Large amounts of historical information to dig through, a high rate of new information being added, contradictions in available information, a low signal-to-noise ratio making it difficult to identify what information is relevant to the decision, or the lack of a method for comparing and processing different kinds of information can all contribute to this effect*".

In order to deal with information overload it is necessary to use a set of techniques and tools that will help us in finding information. This set of techniques and tools are provided by the research field of information retrieval. Again [34] defines information retrieval as:

> "*…the art and science of searching for information in documents, searching for documents themselves, searching for metadata which describes documents, or searching within databases, whether relational stand alone databases or hypertext networked databases such as the Internet or intranets, for text, sound, images or data*"

In a general sense, information retrieval deals with finding information using three different approaches: retrieval approach, filtering approach, and browsing approach. The retrieval approach is based on the idea that the information retrieval system is presented with a query by the user and it should bring out some information resources that are considered to be important or relevant by the user. The second approach is designed to sort through large volumes of information and show the user interesting sources. One important characteristic of information filtering is that the collections are changing constantly by new items added or updated, and the user wants to receive the new information that is useful to his needs. Finally, the last approach is tightly related with the way results are presented to the user. Browsing is frequently used when the user does not know exactly what he wants therefore he explores the available information hoping to find something interesting.

In the next sections we will focus on recommendation systems, which are systems that can adapt the information retrieval process in order to obtain better results. Recommendation systems use one of the information retrieval approaches (retrieval, filtering, or browsing) in conjunction with knowledge about the users, to suggest personalized and interesting items to every user. In this way the recommendation system can produce more personalized results. Below we will explain the most important characteristics of these systems, how they work, and different existing types. Besides, we will define some terms that we will be using in the rest of this document.

2.2 Recommendation systems

Nowadays, there are a lot of recommendation systems, accessible via internet, which attempt to recommend to users several products such as music, movies, books, etc. In order to understand them first it is necessary to have a description. In a general way, recommendation systems are systems which intend to acquire opinions or preferences about items from a community of users, and use those opinions to present other users with items that are interesting to them. From this general description we can see that recommendation systems need two basic things to work properly: Information about the preferences of the users, and a way to determine if an item is interesting for a user. Normally, the users' information includes external information, such as user profiles, purchases histories, and product ratings [7]. The way to determine whether an item is interesting to a user or not, depends on the kind of recommendation system, and in the techniques used to find similarities among items or users. We will discuss more about this later in this chapter.

The above description is quite general and could be applied even to persons that recommend items to other persons (the salesman in a records' store). A more specific definition of recommendation systems is given in [34]

> "*System that produce individualized recommendations as output or have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options*"

The main keywords in this more formal definition are *individualized* and *personalized*. These terms indicate that every user will be presented with different information sources or items depending on the information the system has about every user. In order to continue or discussion about recommendation systems, how do they work, and which kinds exist currently, we will have to define several terms that we will be using through the rest of this document.

2.2.1 Terms and Concepts

The following terms are often used in a recommendation system and the definitions introduced here are based on the work presented in [34].

- Item: in the context of recommendation systems, an item represent the information the system posses about any object. An object can be an electronic document, a product, a person, a service or anything that can be represented by information

- Recommender: a recommender is any entity that gives personalized recommendations as output to users' preferences. It may be possible that a recommender does not produce a specific output, such as a list, but they might guide somehow the users in an individual way to useful or interesting items. A recommender could be a person or a software system.

- Recommendation: this is the output of a recommender; it can be compound by an item or a list of items. The items presented to the users have to be interesting to them, according to the recommender. The criteria used to determine if an item is interesting or not for a user depends exclusively on the technique used by the recommender.

- User's Interest: this is an abstract representation of how much a user appreciates an item. This is a subjective concept and it is hard to represent it in an objective way.

- Prediction: the expected interest of a user in one item. This concept is different to the concept of recommendation. While some systems might present predictions with the actual recommendations, others can produce recommendations only.

- Rating: an objective measure representing a user's interest. The possible values of this measure are given according to a scale established by the designer of the recommendation system.

- Predicted Rating: an objective measure representing the expected interest of a user in certain item. This measure is estimated by the system and its possible values are elements of a specific scale.

- Actual Rating: objective measure representing the real interest of the user in a specific item. This value is given by the user himself according to the scale of rates of the system.

- Prediction Accuracy: a measure that indicates at which extent the predicted rating agrees with the user's actual rating. The more accurate the predictions the better the performance of the recommendation system.

- Prediction Technique: the specific algorithm that the recommendation system will use in order to calculate the predicted rating of an item.

2.3 The Recommendation Process

Once we have defined some concepts and terms generally used in recommendation systems now it is necessary to explain the way these systems work and produce recommendations to the users. In a general way every recommendation system follows a specific process in order to create recommendations. Here we will explain this process and each of the steps involved.

If we see the process of recommendation as a black box, as shown in figure 2.1, we can identify two sources of information needed as input for the process. These sources of information are the users' profiles and the information about items or products. Ideally the information stored in the profiles is related with the preferences of the users and should be given explicitly by the user itself. However, this information can also be extracted from other external sources such as web pages, buying behavior, etc. The

information about the items can range from special metadata of the product, information extracted from the item, or the item itself in the case of electronic documents. In the case of audio or video recommendation systems this information could produce databases of huge dimensions. Also in figure 2.1 we can distinguish that the final product of the system will be a set of recommendations for the user. The final representation of these recommendations depends on the system itself but it may range from ordered lists of items, snapshots of the items, or the whole items.

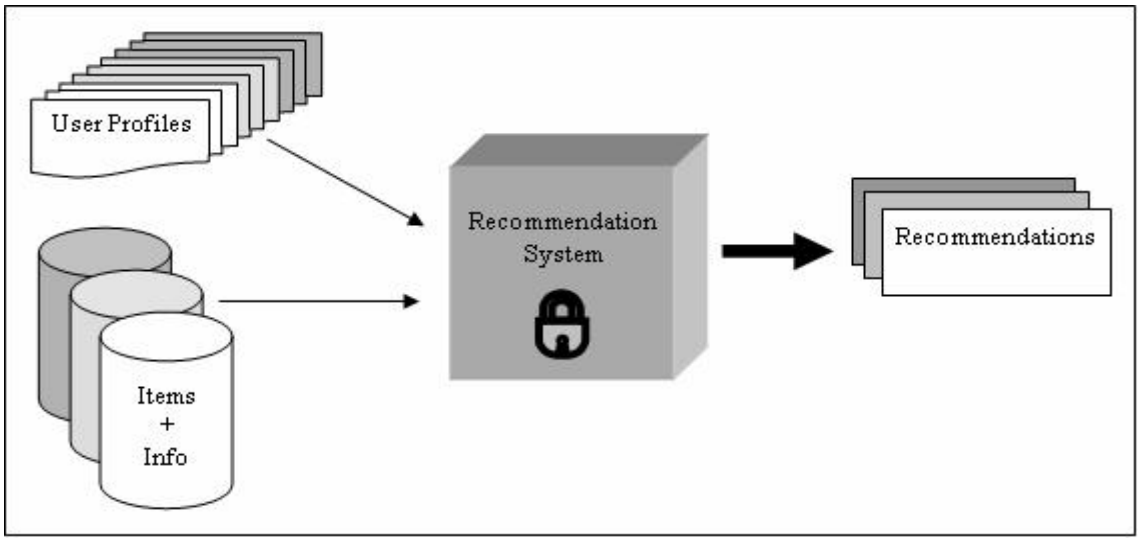

Figure 2.1 Recommendation process as a black box

The recommendation process in a more detailed way is showed in figure 2.2. It includes the following steps: information recollection, selection, transformation, structuring and presentation. From all the steps presented in figure 2.2, the information recollection step is the only one that is not done by the system itself. Below we will give a small description of each of these steps

Information Recollection

Although the information recollection step is not performed by the recommendation system itself, it is a really important step. It includes the recollection of users' personal preferences and information about items such as metadata, features extracted directly, etc. This step has a special importance since it will be the base for the whole recommendation system. If the information collected presents incongruence or contradiction the system will not be able to produce not even regular recommendations. For this reason especial attention should be put in collecting information that truly reflects the preferences of the users, or information that truly represent the items.

Selection

According to [34] the step of selection consists of determining "*which items are interesting or relevant enough for a user and removes all other items from the retrieved set of items*". The way the selection of the items is done depends strictly on the approach taken to find items that are similar to the ones the user consider relevant. Therefore, the key concept in this step is how we define similarity between items. The concept of similarity could be defined in terms of other users' profiles, of features

extracted from the items, or in terms of metadata associated with the items. We will not define the concept of similarity yet, since it depends on the approach taken.

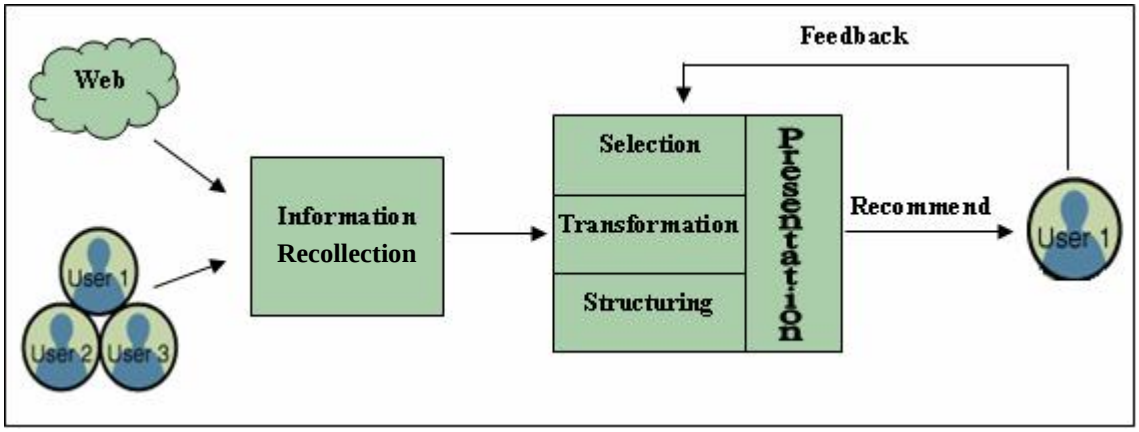

Figure 2.2 Recommendation Process

Transformation

The main objective of the transformation step is to perform some modifications to the items retrieved. This step is only optional in the general recommendation process and it addresses transformations such as summarization, change in the quality of the items, creation of thumbnails or snapshots for its further presentation, etc.

Structuring

The step of structuring is related with the construction and organization of the structure that the user will use to navigate through the different recommended items. This step can include activities such as grouping the items according to certain characteristics, sorting the groups of items, sorting items inside these groups, linking items that have some relationship, etc.

Presentation

The final step in the recommendation process is associated with the presentation of the different retrieved and structured items to the final user. According to [34] it "*deals with issues such as layouts, document formats, colors, fonts, and presentation medium*". This final step in the process should be designed and executed carefully since the user will interact with the system by means of its results.

Feedback

One additional step that might be interesting to consider is the one related with the feedback to the system. Though this step is optional, it can help greatly to improve the results of the recommendation system. With an appropriate feedback any of the previous steps could be improved. The kind of feedback obtained from the user could be of two kinds: implicit or explicit. In explicit feedback the user provides the system with information about how relevant the recommended items are. On the other hand implicit feedback is obtained from the user by analyzing his usage behavior. For instance, how much time he spends in looking at the retrieved items.

3 Classification of Recommendation Systems

Since one of the most important components of every recommendation system is the one in charge of making predictions, it is logical to classify them according to the prediction technique they use. Base on the prediction technique they use it is possible to classify the recommendation systems in two main groups: systems that use social-based prediction techniques and systems that use information-based prediction techniques. Additionally to these two first groups there is a special group of recommendation system. This third group is made by systems that use hybrid methods for recommending items. This new category is emerging in recent years and it has some important features that we will describe in detail in section 3.3.

3.1 Information-based Prediction Techniques

Information-based prediction techniques base their results in the analysis of all the items plus the preferences of the user. They only consider the actual user and do not take into account the information related to other users. Furthermore, these techniques are considered domain specific since they have to analyze the information stored in the item or the metadata associated with them. Some examples of these techniques are: case-based reasoning or content-based, information filtering, attribute-based techniques [34].

The main assumption under case-based reasoning or content-based techniques is that a user has similar preferences over similar items. The more similar the items the more equal the preferences of the user on those items. In information filtering the system is in charge of ordering huge amounts of items and delivering to the user only the items that are relevant for him. Finally, attribute-based techniques estimate their results on the base of the attributes of an item. Each attribute of an item has an importance weight and the overall prediction is calculated taking into account the value of each attribute. Being content-based techniques one of the main focus on this thesis in the next sub-section we will discuss in more detail content-based recommendation systems and the current work on them.

3.1.1 Content-based recommendation techniques

According to [12] these approaches attempt to retrieve useful information from items of the collection. This extracted information should be a good indicator of how much the items are relevant for the user. Besides, extracting features from the items themselves instead of trusting on uses behavior allow recommending new unseen-before items. Although this first description of content-based description techniques is quite general and little bit fuzzy, since it does not explains clearly what is a good indicator of relevance, it already introduces the notion of features extracted from the items and the idea of not relying on user behavior.

The work presented in [7] states that the main assumption under content-based approaches is that an item or document can be identified by a set of features extracted directly from their content. Furthermore it expresses that one premise on these approaches is that "*metadata is either not suitable, or unreliable, or missing*". This

second description is more specific and puts special emphasis in the idea of describing an item as a set of features derived from it.

As explained before in section 2.41, content-based recommendation techniques are a special case of information-based techniques; in that way they attempt to make predictions based on the analysis of the items or the metadata associated with them. In [34] these techniques are described as techniques that "*look at all items a user has rated in the past and determines how similar they are to the current item. For those items that are similar enough, the old ratings are used to calculate a predicted rating for the new item*". This description introduces the notion of similarity among items and also takes into account the idea of previously rated items.

Based on the descriptions given in [7], [12], and [34] we will describe a content-based recommendation technique as one that analyze items in order to extract features that can describe them. Once features are extracted, the system will use them in order to find items that are similar to the user's preferences. One way to represent this user's preferences is by means of rates, previously given by the user. Since these techniques depend on the analysis of the content, the quality of the available data is a determinant factor in the quality of the results produced by the system. For the purpose of this document we will assume that specific metadata associated with the items is not present at all.

The key concepts in the description presented above are *features extracted*, and *items similarity*. Since this document is related with music recommendation systems, we will focus in the next section on features extracted from music items and music similarity. Besides, every time we use the term recommendation system we implicitly refer to a music recommendation system.

3.1.2 Music Similarity

Content-based recommendation methods are normally based on the idea that the system should recommend new items depending on their similarity or dissimilarity with the user's personal preferences. As we can see, the concept of similarity is one fundamental part of the algorithms needed to organize and recommend music. For this reason, in this section we will try to explain music similarity and the theory behind it.

Some theories and principles behind music recommendation based on music similarity, have a motivation in theories on music perception. For instance, the work presented in [36] presents a model that attempts to describe how music processing is performed in the brain by human beings. Furthermore, in [7] it is stated that the enjoyment of music listening can be due to principles of expectation-confirmation and expectation-denial. These principles suggest that similarity in the musical content can be a viable approach to provide users with preciously unknown music that can be enjoyed. For more details about expectation-confirmation and expectation-denial principles please refer to [7]. All these principles and theories made an attempt to understand how humans consider two musical pieces similar.

Although all the theories behind music similarity are based on theories of music perception and psychology, the research area that attempts to deal with music similarity

in more practical situations is the area of Music Information Retrieval (MIR). MIR is an emerging area which main purpose is to fulfill the music information needs of hundreds of persons. In [17] these needs are expressed as practical applications where content-based music recommendation systems could be used. Some examples of these applications are:

- Identify automatically music in stores by humming.
- Help the musicologists find out how composers influenced one another or how their works are related to earlier work.
- Copyright infringement could be resolved.
- Identify music played in bars, radio, etc by recording a sample and using it on a fingerprinting system.
- Records made by security systems can be searched for suspicious sounds.
- Helps content-based video retrieval systems.
- Search sounds similar to a sample in big music libraries.

In order to estimate the similarity or dissimilarity between two musical items it is necessary to compare sets of features which represent the items in a meaningful way. The choice of which features to compare and how to extract them depends mainly in two important aspects: the dimensions of music a user considers important, and on how musical items are represented. In the next paragraphs we will talk about different musical content representations and some of their characteristics.

The work presented in [17] describes three basic representations of music items: common music notation, time-stamped events, and audio. Common music representation is the traditional way of representing music, using scores. This kind of representation gives enough information to perform the music in a good way. Time-stamped events representation finds its best representative in the Musical Instrument Digital Interface (MIDI). MIDI is an industry-standard protocol that defines musical event in one piece of audio. This kind of representation stores information related with the tempo, key signatures, and instruments.

Common music notation is normally used by people with some background in music theory, like professional musicians or musicologist. For this reason, MIR is normally separated in symbolic MIR, where time-stamped events representation is used, and audio MIR, where unstructured audio signals are used [25]. To estimate audio similarity typically symbolic MIR uses features related with melodic information while audio MIR uses features that describe timbre and rhythm. We will discuss more about feature extraction in section 2.5.2.

The work presented in [17] describes several ways to compute music similarity on both symbolic representation, and audio representation. Some methods used in symbolic MIR are: string-based methods for monophonic melodies, geometry-based methods for polyphonic melodies, and probabilistic matching. For instance [14] attempts to extract

from MIDI files sets of weighted points and then compute the similarity between songs using the earths mover's distance. Another work related with symbolic representation is the one presented in [19]. In this work they try to represent music as a string of symbols and then use string-matching methods in order to compute similarities. The work in [27] presents a method for extracting the melody of a polyphonic song from a MIDI file. The melody extracted is represented as a set of chords that will be used later to compute similarity.

Although some promising results in symbolic MIR, one of the major drawbacks of these methods is that they need the symbolic representation of the audio content and of course, this is not the most common way to store audio contents. Although some work has been done in order to automatically create symbolic representation from audio files [20], this problem is far from being solved and presents a lot of troubles especially with polyphonic music.

On the other hand, audio representation attempts to use machine learning techniques to compare set of features. Some examples of these techniques are Earth's movers distance, Self-Organizing Maps, or simply mean and standard deviations of the set of features. The decision of which techniques use to compute music similarity is tightly related with the set of features extracted from the audio files.

To conclude, in the last years a lot of research has been done to find different techniques to estimate music similarity. The most common approaches use symbolic representation or audio representation. Although symbolic representation techniques achieve good results, [35] and [7] agree that these techniques are limited to the amount of music, with this representation, that is available in electronic format nowadays. For this reason, most of the research works on MIR, and the developed systems use audio representations. In this document we will not longer deal with symbolic representation and we will only focus on audio representation and techniques.

3.1.3 Features Extracted from Music

Since content-based recommendation methods are based on detecting the similarity between two items by comparing representative sets of features, it is reasonable to present some of the theory behind feature extraction in audio content. Features extraction from audio representation is the major trend for computing music similarity and it is based on the use of technique from the field of signal processing such as Direct Fourier Transform (DFT) or Mel Frequency Cepstral Coefficients (MFCCs).

In order to extract truly representative features of audio signals, first it is necessary to understand that music is conformed by several facets, each of them playing a different role in the description of music. The publication in [36] is a research in the field of psychology and neuroscience and attempts to provide definitions for aspects that are truly relevant for music. This work states that our auditory system works as a set of interconnected modules that can represent clearly rhythm and meter (temporal organization) as well as tonality and melodic contours (pitch organization). The work presented in [45] defines some important facets of music such as pitch facet, temporal facet, harmonic facet, timbral facet. Additionally to these facets the work in [7] includes some others dimensions of music such as structure, melody, and acoustics. In the next

section we will explain in more detail these facets and some work done to extract features that describe them

Pitch

The pitch facet is related with the perceived quality of the sound, and it is mainly a function of the fundamental frequency. This facet might range from low or deep to high or acute. Some example of features related with pitch dimension is pitch histograms. The idea of pitch histograms was introduced in [25] where there is an explanation on how to compute these histograms from audio signals and then use them in the context of musical genre classification. The main idea of this work was to calculate pitch histograms and then use pattern recognition classifiers to recognize these patterns in songs. For more details about how to extract pitch histograms from audio content please refer to [25].

Rhythm

The temporal facet of music is concerned with the duration of musical events. It conveys information such as tempo indicators, meter, pitch duration, and harmonic duration. Since temporal aspect is determined by complex combinations of tempo, pitch, harmonic durations, etc, it is quite difficult to represent it for retrieving purposes; percussive sounds are good identifiers of temporal facet. The work presented by Lidy in [18] explains how to extract rhythm patterns from audio content using spectral data and different algorithms such as the short time Fast Fourier Transform and others.

Harmony

The harmonic facet is related with the phenomenon that is produced when two or more pitches are produced at the same time; it represents the organization of simultaneous sounds along the axis time. Harmony in an audio content is normally represented by a sequence of chord and the key of the music. Some works like the ones presented in [26] and [28] attempt to estimate the tonality of songs based on the Harmonic Pitch Class profile (HPCP). The HPCP is a vector of low-level features that allows estimating other high-level features such as chords and key of the audio. For more information about HPCP and how to estimate chords or the key of music please refer to [26].

Timbre

The timbral facet is related with the characteristics of sound that allow people to perceive different two sounds with similar pitch and intensity. In order to extract features relates with timbre most the actual systems currently use Mel-frequency cepstrum coefficients (MFCCs). Normally MFCC are used in speech recognition since they provide a concise representation of spectral characteristics. The works presented in [6] and [35] explain how to compute MFCC to obtain a vector of features.

The MFCCs recently have been used a lot in the area of MIR. For instance, works like [4], [8], [23], [29], and [30] extract the MFCCs from the audio signal. With these MFCCs they try to represent timbral texture or the envelope of the signal. The extraction of these features will produce a vector of coefficients that can be used later to measure similarity among songs. Besides, in works like [22] and [24] they made an

attempt to measure music similarity or classify sounds by fitting parametric probability models with these features.

From all the dimensions, timbre is the one that it is used most. The reason for this is that it is believed that users are particularly sensible to timbre and less sensible to other dimension of music. Besides, work presented in [51] states that MFCCs can roughly model some important characteristic of the human auditory system. These characteristics are: the non-linear frequency resolution, the non-linear perception of loudness and at some extent the spectral masking effects

Structure

Finally, structure facet is related with high-level features such as repetitions, interleaving of themes and chorus, etc. According to [7], at current state of the art it is still quite difficult to represent the structure of music in a compact and meaningful form. Nevertheless, if is probable that structure is an important dimension of music and will be need to better understand the concept of music.

Orio in [7] defines timbre and acoustics as short-term facets since they tend to change slowly over time and they could be represented with time-independent snapshots. Rhythm, melody, and harmony are middle term features and can be described as time sequences of notes. On the other hand, structure depends on the organization of short-term and mid-term facets in time; it is a long-term facet of music.

One work that seems especially interesting is the one described in [1]. In this paper Cano is presenting MusicSurfer, which is a system for content-based audio recommendations. The interesting part of this work is the use of several features to represent the audio content. For instance, they compute rhythm from several descriptors such as tempo, meter, rhythm patterns and swing. Besides, they also take into account tonal aspects like tonal strength, key note, key mode, etc; they also consider timbre, which is derived from spectral characteristics of the content, and dynamics of the audio signal, that is the variations of loudness/amplitude with time.

3.1.4 Advantages and disadvantages

According to Cano in [1] techniques based on standard machine learning algorithms and the use of relatively low-level features of music are giving quite promising results in the field of MIR. Nevertheless, content based recommendation methods have some advantages and disadvantages. Here we will explain some advantages of content-based methods and to conclude section 2.5 we will discuss the main disadvantages of them.

Advantages

1. One of the most important advantages of content-based methods is that they can recommend completely new items. For instance, items that have never been rated by any user are hard to recommend by other techniques such as collaborative filtering.

2. Since recommendations of music are based on content directly is it possible to recommend completely new artists.

3. One last advantage is that the content of the music is constant, that means that an item should be analyzed only once; it is not necessary to analyze it again if new items are included in the system or if new information about users is obtained.

Disadvantages

1. It is important to mention that content-based methods do not take into account the opinion of other users to recommend items. This fact can be seen as a disadvantage since normally users' preferences are not unique and the information provided by other users, as we will see on section 3.2, could provide some important information that otherwise is impossible to find out by only analyzing content.

2. Furthermore, content-based methods normally recommend items with similar content to that of user's preferences. This issue can also be seen as a disadvantage since content-based methods can achieve variety in artist but they cannot obtain variety of contents, unless we use the idea of dissimilarity instead of similarity.

3. The last but not the least important disadvantage is that until now is still an open issue how to extract truly relevant and significant features from audio content. Besides it is not clear enough which features to use; some users consider some facets of music more important than others do.

3.2 Social-based Prediction Techniques

Social-based prediction techniques take into account the information about users and their preferences. They analyze not only the information of the actual user but the information of the whole set of users. These techniques do not use the information about the items while recommending them, for this reason we can say that these techniques are domain independent. Some examples of social-based prediction techniques are: collaborative filtering, item-item filtering, stereotypes and demographics, popularity, average.

In collaborative filtering the main idea is based on the assumption that people who rate items in a similar way probably have same preferences and tastes. On the other hand, in item-item filtering the main assumption is that items that have similar rates are probably alike. The use of stereotypes and demographics are related with the fact that people who comes from the same background usually exhibits exchangeable preferences. Typical data that is used to describe the background of a user is age, gender, occupation, education, demographic data, etc. Finally, popularity and average are quite simple prediction techniques; they simply recommend items based on their popularity among all users, or on the average of all the set of ratings of an item. These are only brief overviews of the social-based prediction techniques, but in next sub-sections we will cover in more detail one of the most important of these techniques: collaborative-filtering.

3.2.1 Collaborative filtering recommendation techniques

Collaborative filtering recommendation techniques started to emerge in the decade of the 90's mainly due to the increasingly growth of the WWW and huge amount of information available trough it. As explained earlier in section 2.4.2 these techniques are domain independent and use the information from many users to determine similarity between items. The works presented in [30], [34], and [35] explains that collaborative filtering techniques are developed as an alternative to manual classification or manual recommendations, which requires too much effort to be considered a valid option.

Some researches such as the ones presented in [4] and [47] state that the basic assumption in collaborative techniques is that people's taste are not randomly distributed and they follow certain pattern or trends depending on the user's preferences as well as the social group he belongs to. Furthermore, these patterns could be understood and exploited by managing user's profiles related with the items of the system. A user profile can be described as a collection of items associated with determined rates given by the user. Once the profiles are created, the system calculates the similarity between users or items via standard formulas for computing statistical correlations.

In a general way, collaborative filtering recommendation systems work as follows: first the system collects and maintains information about the user. This information includes specific interest of users in certain items and it is stored in separated profiles. Once all the profiles have been collected the system compares all the profiles in order to determine similarities between them. The way similarity is computed depends on the algorithm used and can vary from system to system. Finally, to produce recommendations to a user the system creates a set with the most similar profiles and use the information contained in this set to do the recommendations.

One of the biggest drawbacks of collaborative filtering is that it is highly dependent on the information provided by the users. If the user does not provide reliable information the performance of the system will decrease considerably. In the past years some work has been done to overcome these situations. For instance, the work presented in [48] attempts to create clusters related with music artists by crawling the web. This kind of information can be used later for testing the reliability of the information provided by users in the system. Furthermore, the work presented in [49] attempts to retrieve collections of lists of related music from the web. Then it uses these lists as pseudo users for collaborative filtering systems. The experiments presented in these works prove that methods for recollecting information automatically are nearly as effective as data provided by real users.

3.2.2 Algorithms for collaborative filtering

One important aspect to consider in collaborative filtering techniques is the way similarity between users' profiles is computed. The work presented in [47] explains several possible algorithms to compute similarity and applies them all to a prototype of

a music recommendation system, named Ringo. Here we will explain briefly the most used algorithm for collaborative filtering. For a more detailed description as well as a more detailed mathematical analysis please refer to [50]

Mean Squared Differences Algorithm

This algorithm estimates the degree of dissimilarity between user's profiles. This is done by calculating the mean squared difference between the profiles. The final predictions then can be done by taking into account all the profiles with degree of dissimilarity below certain threshold *L* and computing a weighted average of the ratings provided by those profiles. The weights used are inverse proportional to the dissimilarity. The formula used for calculating the similarity between two users is given below.

$$D_{xy} = \frac{\sum_{n}^{Na} C_{xn} \times C_{yn} \times (S_{xn} - S_{yn})^2}{\sum_{n}^{Na} C_{xn} \times C_{yn}}, \qquad (3.1)$$

Where:

C_{xn} = [1, 0]: depending whether song *n* is rated by user *x* or not.
C_{yn} = [1, 0]: depending whether song *n* is rated by user y or not.
S_{xn}: is the rate of song *n* given by user *x*.
S_{yn}: is the rate of song *n* given by user *y*.

From the formula 3.1 we can see that the similarity between two users depends only on songs that has been rated by the two users, since the values of C_{xn}, C_{yn} will be 0 if one of the users does not rate the song *n*.

Pearson Algorithm

This algorithm it is an alternate approach to measure similarity between profiles and it is based on the used of the standard *Pearson r* correlation coefficient. The possible values of the Pearson coefficient range from -1 to +1 including 0. Values near -1 indicate a negative correlation while values close +1 indicates a positive correlation; a value of 0 shows no correlation at all. Once Pearson coefficient has been calculated the recommendation can be done as in the previous way by averaging the values of the most similar profiles. One important characteristic of this algorithm is that it takes into account not only positive correlation but also negative correlation to make the predictions. The formula used to calculate the similarity between users takes into account the rates of songs provided by both users and it is defined below.

$$D_{xy} = \frac{\sum_{m} (S_{xm} - \overline{U}_x)(S_{ym} - \overline{U}_y)}{\sqrt{\sum_{m} (S_{xm} - \overline{U}_x)^2 \sum_{m} (S_{ym} - \overline{U}_y)^2}}, \qquad (3.2)$$

Where:

S_{xm}: is the rate of song *m* given by user *x*.
S_{ym}: is the rate of song *m* given by user *y*.

$\overline{U_x}$: is the average rate of user *x*.

$\overline{U_y}$: is the average rate of user *y*.

The formula 3.2 defines the way similarities between users are measured. This formula takes into account only songs that were rated by both users and measures the difference between each common song and the average of the rates of each user. Once all the correlations between users are calculated then the prediction of a specific song can be calculated easily as a weighted average of all the user ratings for this song with the formula 3.3

$$P_{xi} = \overline{U_x} + \sigma_x \frac{\sum_k^{Nu} D_{kx} \times \left(\frac{(S_{ki} - \overline{U_k})}{\sigma_k} \right)}{\sum_k^{Nu} D_{kx}}, \qquad (3.3)$$

Where:

P_{xi}: is the predicted rate for user *x* on item *i*.

S_{ki}: is the rate of song *i* given by user *k*.

D_{kx}: is the correlation between user *k* and user *x*.

$\overline{U_x}$: is the average rate of user *x*.

$\overline{U_k}$: is the average rate of user *k*.

σ_x : is the standard deviation of all the rates of user *x*.

The main difference between this method and mean squared difference is that in this method it is not necessary to select users that are closer to the testing user. This method provides a way to take into account all the users of the set even when they have a negative correlation.

Constrained Pearson r Algorithm

This algorithm was presented in [47] and explained in more detail in [50]. Basically, this is a small modification to the standard *Pearson r* algorithm. It is based on the fact that the scale of ratings is absolute, but values that are below 4 (in the case the scale ranges from 1 to 7) are negative, while values above 4 are considered as positives. In this way the correlation coefficient between two profiles is increased only when there is an instance where both profiles have rated the item positively or both negatively. The main difference with the standard *Pearson r* algorithm is that this version does not take into account the negative correlations. We will not get into more detail of this algorithm since it is based on the same idea of Pearson r algorithm and we will not use it in our experiments.

The experimental results presented in [47] and [50] show that the algorithms described above produce recommendations of good quality. However, these results only consider simple profiles and the algorithms first need a sufficient amount of initial ratings provided by the user in order to work properly. Despite the fact that collaborative filtering recommendation techniques produce some good results and they overcome

some of the troubles found in content-based methods, they also present some troubles and disadvantages by their own. We will discuss about these issues in more detail in the next section.

3.2.3 Advantages and Disadvantages

One of the major advantages of collaborative filtering recommendation techniques is that they overcome some of the limitations presented in content-based techniques. Below we will enumerate them.

Advantages

1. There is not need to have an electronic representation of the items so the computer can parse them.

2. Furthermore, the recommendations produced might be very different, in content, from the original preferences of the user. This means that there is more variety in the kind of music recommended.

3. Besides, according to [47] recommendations are based more in the quality of items, rather than in more objective properties such as content.

4. Finally, collaborative-filtering techniques are domain independent and can work perfectly in domains where it is hard to extract content information from the items, or where there is not content at all associated with the items [10].

Disadvantages

1. According to the works presented in [9], [10], [31], and [34] the new item problem or bootstrapping is one of the most serious problems of these techniques. These problems refers to the situation where a recommendation system does not have enough information about a user or an item, therefore it can not do any recommendation to this user or about this item. In [34] three kinds of bootstrapping problems are explained: new item, new user, and new system. New user problem occurs when a user is new to the system and he has not provided enough information about his personal preferences. New item problem occurs when a new item is introduced into the system; in this case none of the users have an opinion about this item so it is not possible to recommend it. Finally, the new system problem is a bigger combination of the above problems.

2. Other important disadvantages of collaborative filtering techniques are related with the quality of the user profiles and the popularity of the items. For instance, in [35] it is stated that these techniques only work correctly if a reasonable amount of reliable data about user preferences is available online. Furthermore, in [32] it is expressed that collaborative recommendation systems tend to fail when there is few information about preferences or when a user has quite uncommon interests.

3. The other issue explained here is related with popularity of certain items. Recommendations using collaborative techniques may not correspond to actual musical preferences but could be biased by the popularity of a certain item [1].

4. According to [4] collaborative systems produce interesting recommendations only for naïve profiles, they get stuck when using bigger profiles, and they can not handle correctly heterogeneous profiles.

Additionally to the disadvantages described above, which are the most common ones, the literature include some other minor possible problems of these techniques. Although these problems might not be always present we will talk a little bit about them below.

a. Sparsity is another problem in CF, described in [10]; it is related to the fact that most users do not rate all the items in the collection and just a few ones. This produces a user-item rating matrix very sparse which leads to a reduced probability of finding a set of users with many ratings in common.

b. Finally, in [31] the idea of non-transitive associations is presented. This means that if two similar items have never been rated by the same user, their relationship is lost. For this reason those two items cannot be classified into the same neighborhood and this will definitely affect the performance of the system.

3.3 Hybrid recommendation techniques

Although collaborative filtering has been very successful in both research and practice, it presents some major disadvantages. For instance, it can not recommend new items to the users and completely denies any information that could be extracted from contents of item [11]. On the other hand, content-based methods fail in providing as good recommendations as collaborative filtering does. The reason for this is that it is hard to extract really high level meaningful features of music from the audio signals. Hybrid recommendations systems are developed in the recent years as an attempt of overcome the weakness of pure content-based or pure collaborative methods.

The main idea behind hybrid recommendation techniques, as stated in [34], is that "*a combination of algorithms can provide more accurate recommendations than a single algorithm and disadvantages of one algorithm can be overcome by other algorithms*". According to [32] incorporating content into collaborative filtering systems allows increasing the quality of a recommendation system. Besides, when data is too sparse additional content information is a need in order to fit global probabilistic models. The work presented in [9] explains that a method that integrates both ratings and content data enables more accurate recommendations with a richer variety than pure content-based or pure collaborative filtering techniques. In the specific case of music recommendation hybrid systems can exploit the content of the music and the different similarities or dissimilarities among user preferences. This specific combination can be decisive factor for recommending truly relevant items to the user.

In the past years some research has been done in the area of hybrid recommendation. For instance, works like the ones presented in [9], [10], [11], and [31] describe common

approaches for using hybrid recommendation techniques. The work presented in [10] is not specifically related to music but gives some ideas about hybrid systems. This work presents some ideas on how to create a content-boosted collaborative filtering system for movies. The main idea of this work is to convert a sparse user-rating matrix into a full ratings matrix using content data. Although the content-based information in this case is extracted from metadata, the general idea still can be used for the specific purpose of music recommendation. The works presented in [11] and [31] suggests a technique that includes the content of items into a collaborative-filtering system to improve the quality of its predictions and solve the cold start problem. Nevertheless, each of these works differs in the way they combine the algorithms for recommending music. In the next section we will discuss about possible ways for combining them.

3.3.1 Methods for Combining Recommendation Techniques

One of the most challenging parts of hybrid recommendation systems is to decide how to combine the information obtained from the content-based part and the information from the collaborative filtering part. Depending on how this information is combined we will define a small taxonomy of methods for combining recommendation techniques. According to [34] these methods can be classified into weighted, switching, mixed, feature combination, cascade, and feature augmentation. Additionally to these methods the work presented in [32] introduces the idea of the three-way aspect model, which is an attempt to combine results using probabilistic models. This idea is used in [9] to define a new method for combining recommendation techniques. Next we will give a small overview of each of these methods.

Weighted

In this method the final recommendation is done by weighting and combining the prediction of the pure recommendation techniques. The simplest way to combine the initial predictions is to average them all and use the result for the final recommendation. Nevertheless, according to [34] this method can lead sometimes to less accurate recommendation than simply using a pure method. This is normally due to the function used to combine the original predictions. For instance, if statistical central tendency methods, such as mean, median, mode, weighted mean, etc, are used then the estimated value will always be situated between the lowest and the highest initial predictions. If all the initial predictions are to the left or to the right of the actual rating, then the final recommendation will be less accurate. However, more advanced forms to combine predictions can include Bayesian Networks, neural network, or other non-linear functions.

Switching

In this method the system switches between all the pure techniques depending on some criterion. For instance, if one technique is not capable of providing an adequate prediction then the other one is used. In order to use this method adequately the hybrid system should have some knowledge about the pure prediction techniques and when they are able to produce a good result or not. This knowledge might not be trivial and can be really hard to express.

Mixed

In this method no combination is done, instead all the predictions produced by the pure techniques are presented to the user. In this way each item has multiple predictions associated with it. The system that implements this method does not require any knowledge about the pure techniques at all.

Cascade

In this particular case the predictions of one pure technique are refined by the other prediction technique. For example, the first recommendation method might produce a coarse list of predictions which is then refined by the second one. The work presented in [11] explains a system that uses this method. The system first use content-based algorithm to find user with similar preferences and then it uses collaborative filtering techniques to make predictions for those users.

Feature Combination

This method is based on the idea that features generated by a specific prediction technique can be used in other prediction techniques. This method is not easy to implement since the system requires a lot of knowledge about the techniques, the features produces by each technique, and the meaning of these features. The work presented by Stenzel and Kamps in [33] uses this method; it attempts to create a hybrid system by first using collaborative methods to create clusters of songs and then using the features extracted from the content. The idea is to use machine learning techniques to map the content of the song to the clusters created by the collaborative techniques. In this work they use the MARSYAS framework to extract the audio features and the collaborative model is build based on playlists crawled from the web.

Feature Augmentation

This method uses the output of one prediction techniques as an input feature for another technique. One typical example of this is the use of content based techniques to create pseudo ratings of the items and then use these ratings in a collaborative technique as if they were provided by the user.

3-Way Aspect Model

This method was first introduced in [32] and it is based on the idea that substantial user preferences cannot be completely described using observable data. This is because neither ratings nor content reflect totally user preferences. This method attempts to represent unobservable user preferences as a set of latent variables. The work presented in [9] uses an extended version of this method for creating a hybrid music recommendation system that produces some interesting results. In the next paragraphs we will explain this model in more detail.

The 3-way aspect model was first introduced for text information retrieval and it is based on the idea that every document in formed by bags of words. So the probability that a user likes a specific document can be estimated using the probabilities that the words of a document can generate a specific topic which is relevant for the user. In [9]

this idea is extended to the music domain with a "bag of timbres" model. In this way we represent every musical piece in the collection as a set of weights of polyphonic timbres. These polyphonic timbres represent the perceptual sounds of the song, not from individual instruments but from their different combinations.

If we want to compare songs later it is logical to use the same set of polyphonic timbres for all the songs. In order to use the same polyphonic timbres for all the songs, first it is necessary to train a Gaussian mixture of models (GMM). The GMM is trained using the MFCCs not from each piece of music, but from all the pieces of music. This training can be done using techniques such as Expectation-Maximization algorithm or others, and will produce a set of means and covariance that will model the common Gaussians for all the song. In this way we assure that the representation of every song is done using the same set of timbres. In [9] the number of Gaussians estimated is set to 10. Therefore every song in the collection is represented as a set of 10 weights, one per every common Gaussian.

Furthermore, the basic assumption in this method is that collaborative and content-based data can be integrated into a single model using probabilistic generative methods. In this way the generative process for the observed data is explained using a set of latent variables. In the specific case of music these latent variables conceptually represent genres that are not given in advance [55]. Therefore, the musical taste of a specific user is represented as different proportion of the genres which can be estimated as the set of conditional probabilities $\{p(z \mid u) \mid z \in Z\}$ (where z is one of the latent variables). A less formal and intuitive explanation for this model is that every user selects genres, which are not explicitly stated, according to their preferences. Besides, every one of these genres generate by their own songs and polyphonic timbres. One important assumption in this method is that users, songs, and timbres are conditional independent through the latent genres. Figure 3.1 shows this model in a more graphical way.

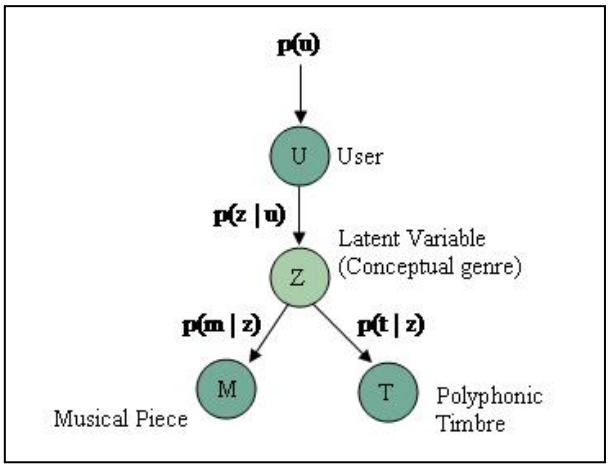

Figure 3.1 Graphical Representation of the 3-Way Aspect Model

Since this model assumes the conditional independence of U, M, T thorough Z we can specify the joint probability distribution p(u, m, t, z) which is given by formula 3.4. In this formula p(u) is the prior probability of user u. Besides, p(u, m, t, z) is the probability that a user *u* will choose genre *z* in his preferences and simultaneously will listen to timbre *t* in piece *m* [9]. If we marginalize *z* from equation 3.4 we obtain equation 3.5 which is the probability that a use *u* picks a song *m* which has timbre *t*. It is

important to notice that the 3-Way aspect model it is a hybrid method since the estimation of the model parameters is done taking into account the contents of the files and the rates of all the users. Once the model parameters are estimated, musical songs can be ranked for a certain user using equation 3.6. For more details on how to estimate the model parameters please refer to [9] and [55]

$$\mathrm{p(u, m, t, z)} = p(u)\,p(z \mid u)\,p(m \mid z)\,p(t \mid z), \tag{3.4}$$

$$\mathrm{p(u, m, t)} = \sum_{z} p(u)\,p(z \mid u)\,p(m \mid z)\,p(t \mid z), \tag{3.5}$$

$$\mathrm{p(m \mid u)}\ \alpha \sum_{\mathrm{t}} \mathrm{p(u, m, t)}\ \alpha \sum_{t,z} p(u)\,p(z \mid u)\,p(m \mid z)\,p(t \mid z), \tag{3.6}$$

Nevertheless this model might produce some good results, it lack efficiency and scalability. According to [55] one of the most important disadvantages of this method is that it is necessary to recomputed the whole model if a user, a song, or even a rate is added to the system. It is obvious that computing the model every single time it is extremely inefficient and the time needed for doing this is proportional to the number of users and the number of items in the system. Furthermore, the amount of memory need for estimate the model is also proportional to the number of users and songs. The work presented in [55] is an attempt to overcome this problems and base its solutions on a method that train the model in an incrementally way. So if one item is included in the collection, it is not necessary to train the whole model but only one part of it. This technique might be a good solution for the scalability and efficiency problem of this method.

4 Evaluation

Evaluation is one of most important parts in any experiment. It is important because it allows us to see whether the results we obtain are good enough or not. Furthermore evaluation will provide us with objective metrics about the real performance of our system. In our specific case we will be able to determine which of the prototypes produces better music recommendations. However, evaluation in music recommendation systems is a difficult task. One of the reasons that make this task difficult is the lack of a common music database in which experiments can be performed. Furthermore, music similarity is such a subjective phenomenon that can vary not only among users, but also across time or according to the mood or context or the context of the user. For this reason it is hard to establish a ground truth that could be used to evaluate systems. Since music similarity is quite subjective first it is necessary to define some ways to measure it. In the next section we will review some related work done before in the area of evaluation.

4.1 Related work on Evaluation

In the last years some effort was put into developing music similarity measures and evaluation techniques of these measures. For instance, the Music Information Retrieval Evaluation eXchange (MIREX) [41] is a good place to find information related with nowadays' similarity measures and evaluation methods. One of these methods is the Evalutron system [42]. Evalutron is a web-based system that evaluates similarity among songs using human raters. Given a specific query and the results produced by many music retrieval systems, a set of persons evaluate the results according to their personal tastes. With the information obtained from the raters it is possible to evaluate the performance of each system individually. This way of evaluating systems requires a lot of effort since it requires the participation of several raters to obtain some useful information. For this reason it is necessary to find some other ways to evaluate systems.

According to [38] music similarity measures basically rely on two aspects of music, acoustic properties, and subjective or "cultural" information. Acoustic approaches analyze the music content directly from the audio signals and can be applied to any music. On the other hand, subjective aspects of music are harder to represent. However, with the growth of the World Wide Web many techniques based on publicly available information are appearing. Many of these techniques include the use of text analysis or collaborative filtering techniques to combine the information from many persons to determine some similarity based on "subjective" information.

In [38] and [39] a methodology is described, for establishing a ground truth which takes into accounts not only audio data but also subjective information gathered from the Web. The main idea of these works is to develop some music similarity measures that take into account subjective information and produce good results. If the measures developed are good enough then they can be used later as a reference for evaluating content-based music similarity measures or other subjective measures. Based on the experiments performed in these papers we will describe below two approaches that produced good music similarity measures.

The *OpenNap peer-to-peer cultural similarity* was introduced in [38] and it is based on the idea that similarity between artists or songs can be inferred from observations. For instance, clusters generated from listening patterns can be taken as a direct measure of cultural similarity and can show relationships that could never be found using edited lists of music. Users' preferences can be used to generate a continuous matrix of similarity. This method receives its name from the OpenNap, a popular music sharing service. In this service it is easy to obtain collections of users preferences. Using these preferences we can say that two artists/songs are similar if they frequently occur together on users' collections. For more details about this similarity measure please refer to [38].

A second approach for producing a good similarity measure is using the so called *Erdos similarity matrix*. This matrix was used for the experiments in [39] and it was produced from data extracted from published music guides, refer to [43] for examples of these guides. This technique is based on the way mathematicians measure their relationships with scientist Paul Erdos; in that sense all the people that have co-authored with him have a value of 1; co-authoring with those authors will earn a person a number of 2, and so on. For our purpose this technique requires lists of similar artists obtained from music guides. For instance, if B and C are contained in the similar-songs list of A then their value will be 1. The similar songs that appear in the lists of B and C will receive a value of 2 with A. Figure 4.1 presents an example of an Erdos Matrix

List of Similar Songs

A → B, C
B → C, D, E
C → E
D → E
E → C

Erdos Matrix of the songs

	A	B	C	D	E
A	0	1	1	2	2
B	1	0	1	1	1
C	1	1	0	2	1
D	2	1	2	0	1
E	2	1	1	1	0

Fig. 4.1 Erdos matrix for a single list of values

The work presented in [9] attempts to create several music recommendation methods and evaluate all of them using the concept of recommendation accuracy. In order to perform recommendation accuracy it is necessary to collect real information about users' preferences. Once we have this information we hide some of these actual rates and check if the recommendations made by the system are in accordance with the hidden rates.

4.2 The Test Dataset

One important component of evaluation is the dataset used to perform the experiments. It is important to use a constant and easy to reproduce dataset. For this reasons we will define here our dataset, the kind of information that we will be using and its sources. Since our goal it is to provide music recommendation using content-based and

collaborative filtering techniques we will need audio content data files and information about music preferences from different people.

The audio dataset will be conformed by the USPOP collection. The USPOP collection [35] is a huge collection of popular artist, albums and songs. It contains 8764 tracks, 706 albums, and 400 artists. This collection was formed taking into account mainly popular artists from the USA and include several versions of some songs, for instance in different live concerts. From this total collection we select all the songs from which we can obtain some users' preferences information. There were specially 53 songs from the whole USPOP collection from which we could not obtain any subjective information. After removing these 53 songs we end with a collection of 8711 tracks, 655 albums and 399 authors. We will include the complete list of tracks, albums, and authors in a separate file (ListSong.dat) for further revision.

Since copyright laws forbid the sharing of audio files, we will not be able to use the audio files directly. Instead we will be using features extracted by other people and shared in electronic form. These feature files do not contain the music itself but only information about some specific features, namely MFCCs. Given these feature files, it is not possible to create the original audio file since a lot of information was missed in the process of extraction. The collection of features files corresponding to the USPOP collection was generously donated by Dan Ellis, professor at the Columbia University in the city of New York, USA.

The information about users' preferences was obtained from Yahoo [44] web site. This web site stores information related to music preferences from millions of songs and users. We crawled this page in order to get information from approximately 10.000 users. It is important to notice that we did not extract personal information from the users, such as names, e-mails, age, etc. We only focused on musical tastes and preferences. We choose these 10.000 users in such a way that each of them has at least a high rate over one track, one album or one artist of the full collection. We did this selection in the following way: for every song in the collection we took one user with a high rate on this song, for every album on the collection we took 3 users with a high rate on this album, and finally for every artist we took 5 users with a high rate on the artist.

From this big set of users we select all the users which have at least 80 rates over the USPOP collection, approximately 1% of the collection's size, and discard the rest of the users. By selection these users we assure that the rating matrix of users and songs is sparse enough and every user in the collection will have enough rates so the system can make a recommendation. At the end of this selection our final set contains 8409 users with an average of 350 rates per user. The total set of user's preferences is conformed by near 3 million rates. The rates were given based on a scale from 0 to 100, being 0 the lowest and 100 the highest. The default value for items that were not rated is -1.

4.3 Evaluation metrics

In section 4.1 we have explained some methods available for evaluating music similarity with subjective data, here we will explain the kinds of evaluation that we intend to use to check the results produced by our prototypes. The evaluation of the

results will be evaluated basically taking into account 3 main aspects: recommendation accuracy, artist variety, and finally total coverage. We will explain these aspects below.

4.3.1 Recommendation Accuracy

Evaluation by recommendation accuracy will be done in the following way. First we will construct a big rating matrix of users and songs. This matrix will contain the rates of every user to every song in the collection (being -1 the default value if the user has not rated the song). Once we have this big users-songs matrix, we will hide randomly 10% of actual rates and later we will use 10-fold cross validation to see if the values calculated by our prototypes are in accordance with the real hidden values.

The work presented in [34] states that "*measuring accuracy is necessarily limited to a metric that computes the difference between the predicted rating and the true rating*". The most used metric to measure prediction accuracy is the Mean Absolute Error (MAE). This metric produces the average absolute deviation between the predicted ratings and the real ones. The recommendation system will be more accurate if the mean absolute error is lower. To compute the MAE the system will use the following formula described in [34]:

$$mae = \frac{\sum_{i=1}^{n} |Pi - Ri|}{n}, \tag{4.1}$$

Where:

P_i: is the predicted rate for item i.
R_i: is the real rate of item i.
n: is the total number of predicted items.

One of the benefits of mean absolute error is that it presents well defined statistical properties. These properties make it possible to test differences in mean absolute error between two systems. Although *MAE* suits very well as a metric for recommendation accuracy, it can hide some troubles in some recommendation systems. For this reason, *MAE* can produce false expectations about the accuracy of these systems. Below there is a list of some of the characteristics of *MAE* that could lead to concealed troubles:

1. If an item do not have a rating, it is impossible to use that item for estimating the value of *MAE*.

2. Errors in high rates or in low rates have the same impact on the *MAE*.

3. The *MAE* may show a good accuracy for system that produce very accurately predicted item with low or average ratings but cannot predict correctly items with high ratings. In this case the system will not be able to produce a good recommendations to the user despite its *MAE* shows a good performance.

4. *MAE* cannot be used to measure the accuracy of techniques that only produces a list of recommendations instead of a predicted rating for every item in the collection.

From the above problems, the first one is related to any kind of recommendation system in general. The second and the third problems are related to systems that attempt to retrieve top-N lists of recommendations. Since our experiments will produce a rating for every item in the collection, these problems will not affect to the metrics of our recommendation systems.

4.3.2 Artist Variety

The second kind of evaluation that we will use in our experiments is evaluation by artist variety. This method is quite simple and takes into account all the previous artists rated by a user. The prototype that gives more recommendations from different artists rather than from already rated artists will obtain a better qualification. We will measure this by taking the top-100 recommendations for every user and calculate the number of total artist in this list and the number of new artist. Then we will take the average of this numbers over all the users and check them. The higher values for the averages the better the artist variety recommendation of the system.

4.3.3 Total Coverage

The last but not least metric to measure the performance of our prototypes will be total coverage. The purpose of this metric is to determine the percentage of songs the system is able to give a recommendation. The calculation of this metric will be done taking into account the total number of items the system was asked to predict, and the total number of items the system could not predict due to lack of information or some other problems. The formula 4.2 was used to measure this metric.

$$T\,\mathrm{cov} = \frac{n - \sum_{i=1}^{n} C_i}{n}, \tag{4.2}$$

Where:

C_i: [0, 1] if the system could make a prediction on item i or not.
n: is the total number of items for which a prediction was asked.

To conclude this chapter, at the end of the experiments each prototype will be evaluated independently according to the aspects described above. Once the three prototypes were evaluated we will perform comparisons and statistics among the three prototypes to see if our hypothesis, that hybrid recommendation systems outperform pure content-based and pure collaborative filtering systems, is probed.

5 Experiments

The purpose of this chapter is to give an overview and explanation of the general setup used for the experiments, the experiments themselves, the development of the prototypes and some important aspects of them such as tuning parameters. The general structure of this chapter is stated as follows: in the first part we will talk about the general setup used to develop our prototypes; the second part of this chapter will explain the content based prototype and some point of interest while developing it; third part of this chapter will discuss about the implementation of the collaborative filtering prototype and some tuning parameters used in its construction; the hybrid recommendation method will be covered in fourth part and finally to conclude this chapter we will explain the experiments done with all of the prototypes.

5.1 General Setup

The general setup for all the prototypes and experiments performed in this thesis was build up using Java technology and a MySQL database server. Java technology was used to implement the prototypes and predict the recommendations for the items. MySQL database server was used to store all the information about the users, songs, personal preferences, song's similarities (in the case of content-based experiments), users' correlations (in the case of collaborative filtering), and results. Additionally, MatLab was used at some point in order to perform some heavy calculations, but we will discuss this in section 5.2 when explaining content-based experiments. In a general way, the database schema is compound by 5 tables: users, songs, albums, artists, rates. Every table stores different information that will be used during the experiments and they are provided with their own indexes in order to execute the different SQL commands efficiently. For more details about the database schema and the relations between different tables please check appendix A.

In order to perform the experiments coherently we need a set of constant data that we will be using for the experiments. For this reason we decided to choose 100 users (evaluation users) from the complete set of users based on the number of rates they have in the whole collection. The number of rates of the chosen users range from 950 to 2500 rates, and they have an average of 1200 rates per user. This sub-set of users gave us approximately 120.000 rates that we used for evaluation purpose. Once we had selected the evaluation users we saved their IDs in a separate file for consistency, so we will be using the same sub-set of users for all the experiments.

In order to evaluate the prototypes properly we did 10-fold cross validation with the rates of the evaluation users. This means that we took the whole set of evaluation users' rates and split it 10 times. Every time that we split this set of rates we took randomly the 10% of the rates of every user and store them in a table with the same schema as the original rates tables. The other 90% of the rates sub-set was stored in a training table that was used mainly for the content-based experiments. Since we desire to have a uniform distribution of the rates in the evaluation table, for every evaluation user we split his rates into groups according to their real rates and take the 10% of every group for evaluation. In this way we assure that the evaluation set will contain at least one song from every group of rates.

Finally it is important to indicate that we also developed two naïve recommendation methods for comparison. One of these methods simply assigns a random value as a prediction for an item. This method is the simplest one and it is expected to produce the worst results from all the prototypes. The second naïve approach uses the average of each user's rates in order to predict the rate of some unrated items. These two methods do not take into account any knowledge about the user or the songs so we expect them to perform poorly in the experiments.

5.2 Content Based Prototype

The implementation of the content based prototype relies on the fact that given the MFCCs of a specific song, it is possible to model this song using a Gaussian distribution of the MFCCs. As explained in section 4.2 the MFCCs files for the full USPOP collection was donated by Dan Ellis and includes a single file containing the MFCC information for every song in the collection. With the help of MatLab software we were able to process each of these files and extract the coefficients themselves. Once extracted the coefficients they are stored as a big matrix of 20 rows and the number of columns is determined by the length of the full song. The next step in the process of modeling songs as a Gaussian distribution consists of extracting the mean and the covariance of the MFCC matrix. The mean is stored in a matrix of 20 rows by 1 column while the covariance is stored in a matrix of 20 rows by 20 columns. Figure 5.1 shows this in a more graphical way. Once calculated, the mean and covariance matrices are enough to modeling the Gaussian distribution.

Once all the models for each one of the songs in the collection are computed we estimate the similarity between these models. In order to calculate the similarity between models we use the Kullback-Leibler divergence. This divergence is used in frequently in probability theory to measure the difference between two probability distributions. Nevertheless, the divergence between probability A and probability B is different from the divergence between probability B and probability A. For this reason Kullback-Leibler divergence cannot be considered as a distance measure by itself. In order to avoid this problem we calculate the similarity between two songs by adding the Kullback-Leibler divergence in both directions (from A to B and from B to A).

Finally, following the advice given in [51] we rescaled the distance obtained. This rescaling step is done in order to reduce the ratio between the max and median values of all the distances. Without rescaling this ratio could reach the order of 12 magnitudes, while rescaling the distances reduce the ratio to 10. Another reason for rescaling the distances as stated in [51] is to improve the recommendation results when combining the spectral information extracted from MFCCs with other information such as fluctuation patterns or others. The formulas used for estimating the distance and rescaling them were extracted from [51] and are stated in equations 5.1 and 5.2. These formulas can be used easily with the MatLab software. The final values for the rescaled distances will range theoretically from 0 to -1 being -1 completely similar and 0 completely dissimilar. In practice or values range from near to 0 to -0.92 and they have an average value of -0.8435.

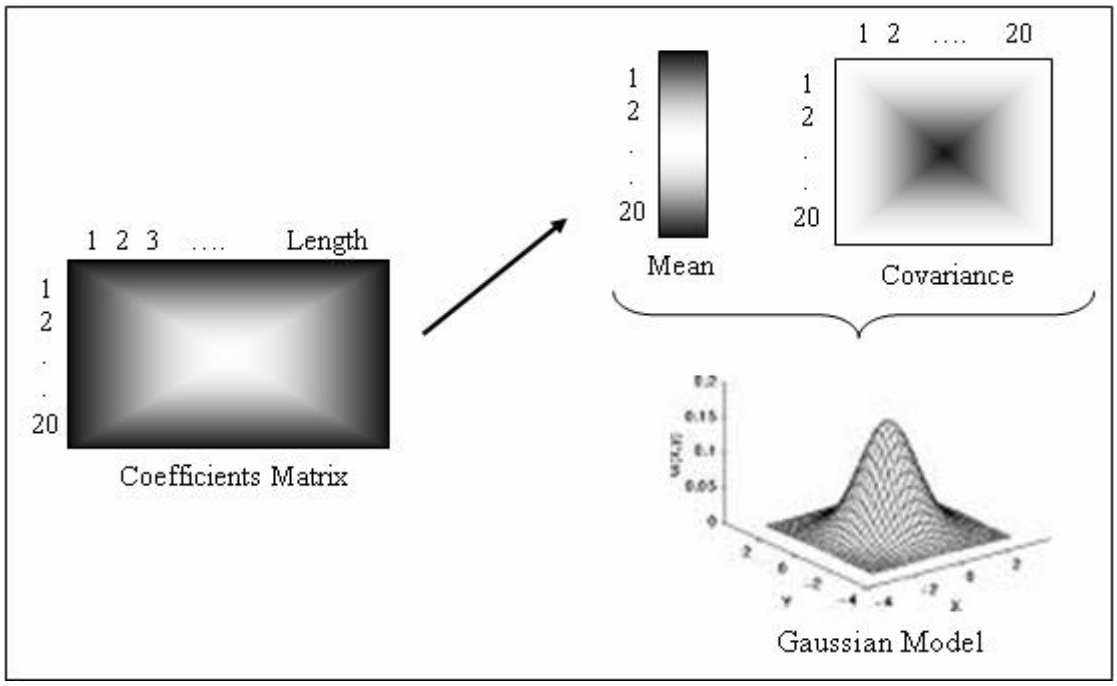

Figure 5.1 Relation between the coefficient's matrix and the Gaussian model

$$D = trace(co1 * ico2) + trace(co2 * ico1) + ... \\ ... + trace((ico1 + ico2)(m1 - m2)(mi - m2)')' \quad (5.1)$$

Where:

co1: is the covariance of the first model.
co2: is the covariance of the second model.
ico1: is the inverse covariance of the first model.
ico2: is the inverse covariance of the second model.
m1: is the mean of the first model.
m2: is the mean of the second model.

Once all the rescaled distances were calculated, we store them in a relational table with a clustered BTree index for fast and efficient access. It is important to notice that we could not estimate the distance for 3 specific songs (3 out of 8764). The reason for this is that the MFCCs files for these songs were corrupted and they didn't store the true coefficients of the songs. For these 3 songs the prototype will not be able to make any prediction and a predicted value of -1 will be returned by the system.

$$Dres = -\exp(\frac{-1}{fact * D}), \quad (5.2)$$

Where:

fact: is the factor used for rescaling, 450 in our experiments.
D: is the original distance.

One interesting problem that we found while developing the content based prototype is related with the fact that we are using two sources of information, namely the Yahoo web site for the rates of the songs, and the MFCCs files for the information about the

content of each song. In order to make the recommendations properly it is necessary to relate this two sources of information; this means that for every song in the collection we have to find out the Yahoo ID that identify this song so there will not be confusion about the rates of a specific song and its content. We solve this problem by developing an algorithm that relates both information sources by checking the song's name, the name of the album, and the artist's name.

All the steps described above can be seen as pre-processing steps than can be executed offline. These steps are common for all the experiments and only need to be executed once. In the next paragraphs we will describe the algorithm itself for predicting a song's rate. The basic idea of the content based prediction algorithm is that in order to make a prediction we only require the information about the preferences of the user we are making the recommendation to. So the first step in this algorithm is to obtain all the rates provided by the user. Then we group these ratings in 11 groups according to their value (0 to 5, 6 to 15… 85 to 95, 96 to 100).

Once we have the set of groups of all the rates made by the user we retrieve the similarities between each of the rated songs and the testing song. For every group we choose the song that it is most similar to our testing song. The final step is to give the recommendation based on the maximum similarity between the testing song and the each of the selected song per group. Although using the maximum similarity is the easiest way to make a prediction, the work presented in [24] shows that this method outperforms other methods such as estimating the average or median distance to the group, or even when representing the whole group of songs as a unique model using k-means techniques.

Another point that it might be interesting of discuss is the one related with dissimilarity. For instance, if the testing song is not similar enough to any of the preferred songs then we should use dissimilarity to make the prediction. In these cases we will find the most dissimilar song and predict a rate of 100 minus the rate of the group. The algorithm presented here has a quite stable performance and can be processed online easily. On average this algorithm can make 100 predictions on 25 seconds, which are around 4 predictions per second. Listing 5.1 show the pseudo code for this algorithm.

```
PredictRate (integer UserID, integer TestSongID) {

        PreferredSongs = GetPreferredSongs (UserID);
        PreferredGroups = GroupPreferredSong (PreferredSongs);
        Similarities = GetSongsSimilarities (TestSongID, PreferredGroups);
        MaxSimilarities = GetMaxSimilaritiesPerGroup (Similarities);
        MaxSimilarity = GetMaxSimilarity (MaxSimilarities);

        If (MaxSimilarity < Threshold) {
        MinSimilarities = GetMinSimilaritiesPerGroup (Similarities);
        MinSimilarity = GetMinSimilarity (MinSimilarities);
        }

        Return RateOfMinSimilarity (MinSimilarity);
}
```

Listing 5.1 Content-Based Algorithm for Predicting Rates.

5.3 Collaborative Filtering Prototype

The key concept in the development of our collaborative filtering prototype is the technique used in order to estimate the different correlations among users. We computed these correlations using the Pearson r correlation, explained in section 3.2.2. According to [34] this technique might present troubles if the number of common songs between users is below certain threshold. We set empirically our threshold to 40 taking into account that it is 0.5% of the total amount of songs and that all the users have at least 80 rates in total. In this way we compute the correlations between every user in the evaluation set and the rest of the users, in the whole collection, which have at least 40 songs in common.

Nevertheless, this strategy reduced considerably the number of stored correlations that will be used to make the predictions. From a total of 871.000 possible correlations, we stored only 730.600. There is one possible optimization for this situation. This optimization consists on adjusting the original correlations between users with few numbers of common songs. The adjustment consists on reducing linearly the original Pearson coefficient. The original coefficient is multiplied by the ratio between the number of common songs and the established threshold. Although this method might increase a little bit the overall performance of collaborative filtering techniques, we did not implement it. The reason for this is that we are considering correlations between users with really few songs in common (less than 0.5% of the total collection) as correlations that will not contribute with real information to the prediction process.

Finally, once all the correlations were calculated we store them in a relational database table with a BTree index that allows fast and efficient access to the records. Once we have all the correlations between users estimated and stored in the database, the next step is to make predictions for the users. Here we will discuss briefly the algorithm used for this purpose. In order to make a prediction for a specific song and a specific user, first it is necessary to know all the users who have rated this song. The second step is to retrieve the correlations between our testing user and every user in the set of users who rated the song. From this set of correlations we can estimate a prediction for the song using formula 3.3. Listing 5.2 presents this algorithm as a pseudo code.

It is important to notice that in order to obtain coherent values it is necessary to use only users who have a strong correlation, whether positive or negative, with our test user. We set empirically the value of the threshold for a strong correlation in 0.1. In this way, we will be using for the recommendation only users that have a correlation bigger than 0.1 or smaller than -0.1. Another interesting point to discuss about our collaborative-filtering prototype is the one discussed also in [34]. In this work Van Setten explains that it is difficult to produce recommendations for users which have fewer rates than a threshold of 50 rates. Since in our prototype we are working only with users that have at least 80 rates each, the situation explained by Van Setten will not affect us.

The paragraphs above describe the general way in which the collaborative filtering was implemented. Since this first prototype is based on the idea of correlation between users, if the set of evaluation users doesn't have strong correlations with other users this method will probably fail. For this reason we have decided to implement also a second collaborative filtering prototype. This second prototype will be based on the idea of correlation between items instead.

```
PredictRate (integer UserID, integer TestSongID) {

    SongUsers = GetUsersOfSong (TestSongID);
    CorrelationSet = GetCorrelationsFromUser (UserID, SongUsers);

    For each (Correlation in CorrelationSet)
        If (Correlation > Threshold) {
            FinalCorrelationSet += Correlation;
        }
    }
    Rate = CalculateRateFromSet (FinalCorrelationSet);
    Return Rate;
}
```

Listing 5.2 Collaborative Filtering Algorithm for Predicting a Rate.

Our second collaborative filtering prototype works in a similar way that the first one, but we have included some modifications in order to obtain better results. First of all, in order to estimate the correlations between songs we use again Person r equation but with a small modification. Instead of looking for all the common songs between two users, we look for all the users who have rates on the two songs. Equation 5.3 shows the modified Person-r formula for computing the correlation between two songs.

$$D_{xy} = \frac{\sum_u (S_{ux} - \overline{S}_x)(S_{uy} - \overline{S}_y)}{\sqrt{\sum_u (S_{ux} - \overline{S}_x)^2 \sum_u (S_{uy} - \overline{S}_y)^2}}, \qquad (5.3)$$

Where:

S_{ux}: is the rate of song x given by user u.
S_{uy}: is the rate of song y given by user u.
$\overline{S_x}$: is the average rate of song x.
$\overline{S_y}$: is the average rate of song y.

With the set of songs' correlations saved in a table with a BTree index for fast and efficient access, we compute the predictions of a specific song and user using the formula 5.4 as stated in [52]. One important indicator of the validity of this kind of prediction is the number of correlated songs that the users have. The bigger this number is the better the prediction for the song gets.

$$P_{u,k} = \frac{\sum_k^{Ns} S_{i,k} \times R_{u,k}}{\sum_k^{Ns} S_{i,k}}, \qquad (5.4)$$

Where:

$S_{i,k}$: is the similarity between songs i and k.
$R_{u,k}$: is the rate of song k given by user u.

Furthermore, we also used the improvement discussed in [34] and explained above. This improvement is related with the number of users that have rated both songs. If this number is below the threshold (40 in our case) we decrease the Pearson r coefficient linearly as explained above. Another improvement made in this second prototype is related with the thresholds for strong correlations. Since we will be able to obtain more correlations we could set the threshold higher than in the first prototype. Having strong correlations, the predictions made by this prototype are expected to be better that the ones produced by the first prototype. Empirically we set this threshold to 0.4 and -0.4.

5.4 Hybrid Recommendation Prototypes

In order to prove or main hypothesis that hybrid recommendation systems outperform pure ones, and they can also solve some of the problems presented in content-based or collaborative filtering, we have developed a simple hybrid recommendation prototype. This prototype is a model that takes into account the predictions made by the pure recommendation prototypes and estimate a new prediction from these values.

This method is the simplest hybrid recommendation method possible. It belongs to the category of weighted hybrid recommendation systems and makes its predictions based on the predictions done by pure methods. This prototype only takes the values predicted by the pure content-based and the pure collaborative filtering methods, and uses as a prediction the average of both methods. This method might perform better than the pure methods if the real rate is in the middle of the predictions made by the content based and collaborative filtering prototypes. However, if the real value is in right or left of both predictions, this method will perform worst than one of pure methods. It is important to notice that in no case this method can perform worst than the two pure methods.

Another important point to discuss is related with the fact that this method depends on the total coverage of the pure ones. In this way if one of the pure methods is not able to estimate a big amount of predictions the hybrid method will be using only the predictions made by the other. Besides, if there are songs for which none of the pure methods is able to make a prediction then this method will also file. In chapter 6 we will discuss these situations in more detail and explain one possible way of solving them.

5.5 Experiments

Based on the metrics stated in chapter 4 we have devised two experiments. The first experiment is related with recommendation accuracy metric while the second one is related with artist variety. Total coverage metric will be used for both experiments to check the overall performance. For the first experiment we used the 10 fold cross validation matrices explained in section 5.1. For every evaluation user we attempt to predict the rates of the 10% matrix using the other 90%. We repeat this process 10 times one per every fold of the experiment. At the end of this experiment we predicted 10 times 12.728 rates, which is an amount big enough to let us obtain realistic statistics.

From the results obtained in this experiment we will compute the MAE in three different levels: per evaluation user, per fold, and a general MAE from the whole experiment. It is also important to notice that the result of this experiment is not a top-N list. Since we were predicting rates for every song in the evaluation set, which contains songs from all the rates groups, the characteristics of MAE that could lead to concealed problems will not apply to these results.

The second experiments were performed using the complete set of songs and the set of evaluation users. For every user in the evaluation set we attempt to predict a rate for all the unrated songs. After the experiment is completed we had 800.000 predicted rates, and we will extract top-100 lists for every user in the evaluation set. Since the final results of this experiment are top-100 lists, it is not possible to use MAE as an evaluation metric. For this reason we will be using artist variety instead. Again, the high number of predicted rates that we obtained here will let us obtain realistic information about the performance of the system.

6 Results

The results presented in this chapter are given according to our evaluation measures and the experiments performed. We have performed two experiments for each of the prototypes we have implemented. The first experiment is related with recommendation accuracy and the second one with artist variety. Additionally we compute the total coverage of each method for both of the experiments and present some computational times for making predictions.

Before explaining the results is also important to state the number of methods that we compared. In a general way we made experiments with the following methods: content based (CB), collaborative filtering with users' correlations (CF1), collaborative filtering with songs' correlations (CF2), and average hybrid methods one for every collaborative filtering method (HA1 and HA2). Finally we also present the results for our naïve methods that give random predictions (RN) and the average of the rates per user (AV).

Furthermore, for each of our collaborative methods we made two versions. The first versions (CF1.1 and CF2.1) give a prediction of -1 for all the songs for which the method fails. In this first version we omit these songs in the computation of our metrics. The second versions (CF1.2 and CF2.2) are a little bit more sophisticated than the previous one. When the method fails the prediction is the average of rates of the user (in the case of using users' correlations) or the average of rates of the song (in the case of using songs' correlations). In these second versions we did not omit these songs for the estimation of the metrics. Since these two versions will have a direct effect on the average hybrid method, we will also present two results for it, one without the failed songs and one with the failed songs (HA1.1, HA1.2, HA2.1, and HA2.2).

6.1 Recommendation Accuracy

Our first evaluation metric is related with recommendation accuracy, in that sense, we divided the results obtained in the experiments in three levels: accuracy recommendation per user, per fold, and a general evaluation. For each of these levels we will compute values for the mean absolute error and compare the methods to see which one performs better. In the level of user we estimated MAEs for every user in our evaluation test. At the level of fold we obtain MAEs for every fold in our 10-fold cross validation experiment. Finally the total MAE was computed from the whole set of users and folds in the experiments.

The results presented in table 6.1 shows the different MAEs for the first 10 evaluation users using all the different methods. We can see that for most of these users content-based method outperforms the rest of the methods. Besides, random and average methods are the ones with lowest performance in most of the cases as was expected. One thing that it is important to notice is the differences between the performances of CF1.1 and CF1.2. CF1.1 apparently performs much better than CF1.2, but indeed for CF1.1 we are not taking into account all songs for which the method fails. We will see in section 6.3 that for CF1 the number of failed songs is a big percentage of the total and this fact might lead to a false appreciation of the performance of this method.

User ID	MAEs[1]										
	AV	RN	CB	CF1.1	HA1.1	CF1.2	HA1.2	CF2.1	HA2.1	CF2.2	HA2.2
1136665141	29.835	38.619	19.216	18.406	20.634	28.757	22.171	26.955	22.257	27.843	22.160
1015814370	19.552	46.313	10.431	8.604	12.172	22.576	15.086	14.545	13.160	16.709	12.784
1148413913	24.008	36.762	13.852	25.237	18.065	27.311	18.606	19.331	15.618	21.049	16.131
1226875802	13.580	51.474	11.520	10.884	13.635	17.603	14.460	9.963	10.981	13.018	12.087
1136271492	6.2985	44.875	4.527	7.223	6.313	9.034	6.577	5.199	5.248	8.726	6.283
1141237820	29.512	39.847	18.578	24.055	22.172	35.934	24.020	24.096	21.619	28.081	21.096
1147739909	20.020	41.701	12.474	9.510	13.896	22.891	16.020	14.128	12.778	15.335	12.716
1027749191	4.8437	49.322	2.708	4.500	4.234	7.343	5.135	4.401	3.786	7.588	5.088
1142084866	27.094	47.172	21.099	23.727	24.921	34.125	26.664	19.617	21.047	23.277	21.172
1144310315	24.301	51.774	16.559	16.715	20.919	27.736	22.102	16.607	16.930	21.634	18.940

Table 6.1 MAEs for the first 10 users in the evaluation set

Fold #	MAEs										
	AV	RN	CB	CF1.1	HA1.1	CF1.2	HA1.2	CF2.1	HA2.1	CF2.2	HA2.2
1	24.298	42.401	18.319	19.905	20.208	26.089	20.858	18.788	19.224	23.078	19.583
2	24.298	42.923	18.031	19.954	20.043	25.976	20.687	19.069	19.112	23.658	19.662
3	24.298	42.629	18.196	20.297	20.334	26.461	20.973	19.202	19.216	23.406	19.661
4	24.298	42.536	17.594	20.118	19.870	26.268	20.625	19.003	18.853	23.444	19.429
5	24.298	42.565	18.040	19.647	19.988	25.938	20.612	18.866	19.018	23.318	19.557
6	24.298	42.652	18.266	19.881	20.108	26.001	20.753	18.786	19.230	23.441	19.717
7	24.298	41.961	17.964	19.970	20.102	26.119	20.703	18.966	18.966	23.429	19.489
8	24.298	42.997	17.864	20.021	19.878	26.201	20.636	18.940	19.061	23.383	19.482
9	24.298	42.565	18.015	20.192	20.087	26.029	20.676	18.811	19.004	23.433	19.585
10	24.298	42.776	18.023	20.273	20.248	26.353	20.884	18.849	19.186	23.355	19.584

Table 6.2 MAEs for every fold in the experiment

[1] CB: Content-Based.
CF1.1 and CF1.2: Collaborative Filtering with users correlations.
HA1.1: Hybrid Average method between CB and CF1.1.
HA1.2: Hybrid Average method between CB and CF1.2.
CF2.1 and CF2.2: Collaborative Filtering with songs correlations.
HA2.1 and H2.2: Hybrid Average method between CB and CF2.1. and CF2.2

Table 6.2 shows the performance of the methods per each fold in the experiment. This MAE is an aggregation of the MAEs of every user in each of the folds. This table shows us more clearly that content-based method is the one that performs better than the other ones, besides random prediction is the one that performs worst as expected. Although collaborative-filtering performs a little bit worst than content-based, it also presents better results than simply average or random methods. Also it is important to notice the performances of HA1.1 and HA1.2. In these cases HA1.1 apparently performs worst than both CB and CF1.1, but as explained before CF1.1 does not take into account failed songs which increases considerably its real performance. On the other hand, the performance of HA1.2 improves considerable the one of CF1.2, which takes into account the failed songs.

It is interesting to notice that the MAE for the average method (AV) is the same for all the folds and for the overall evaluation. The reason for this is that this method always predicts the same value for a particular user, which is the average of its rates. Since every fold contains the same number of users and the same name of songs per user, the MAE will be always the same. Besides we can see that AV method outperforms CF1.2 which it was expected only partially. In the next paragraph we will explain in more detail the reasons for such a poor performance in CF1.2

Although CF1.1 presents a relatively good performance, we were expecting to obtain better results from it. The main reason for this relatively low performance is due to the correlations between users. For most of the users' preferences downloaded from internet we had really low correlations between them. Figure 6.1 presents a graph of the distributions of the correlations between our users evaluation set and the rest of the users. We can see that the mean of this distribution is around 0.022 which is a really low correlation. Given such low correlations between users, most of the predictions done by the system have values bigger than 100 or values smaller than 0. For this reason we attempt to solve this problem by given a final value of 100 to all the predictions with values bigger than 100, and all the predictions with values smaller than 0 are assigned a final value of 0. Despite this improvement we can see that in general CF1 is not an accurate method in this situation. Nevertheless, you should notice that performance of CF1.1, which does not include failed songs, it is really good, but the amount of failed songs is near to 25%. We will discuss this situation in more detail in section 6.3.

Furthermore, we can see from tables 6.1 and 6.2 that CF2 outperforms CF1 as anticipated. This is due to the fact that using songs' correlations instead of users' correlations allows more data from which estimate more accurate predictions. Besides, the number of failed songs, as we will see in section 6.3, in CF2 is drastically reduced in comparison with CF1. One aspect of collaborative filtering that we should point out is the one related with the direct relation between the number of failed songs and the threshold for strong correlations. In the case of CF2 we notice that setting the threshold to 0.3 and -0.3 allow us to predict 99% of the songs but we obtain an overall MAE of around 22.9. However, if we set the value of the threshold to 0.4 and -0.4 we obtain 90% of successfully predicted songs but we also decrease the overall MAE to 18.9 which is quite close to the value of CB. This fact might lead us to think that with the enough amounts of data and a threshold above 0.5 and -0.5 collaborative filtering methods might outperform easily content-based ones.

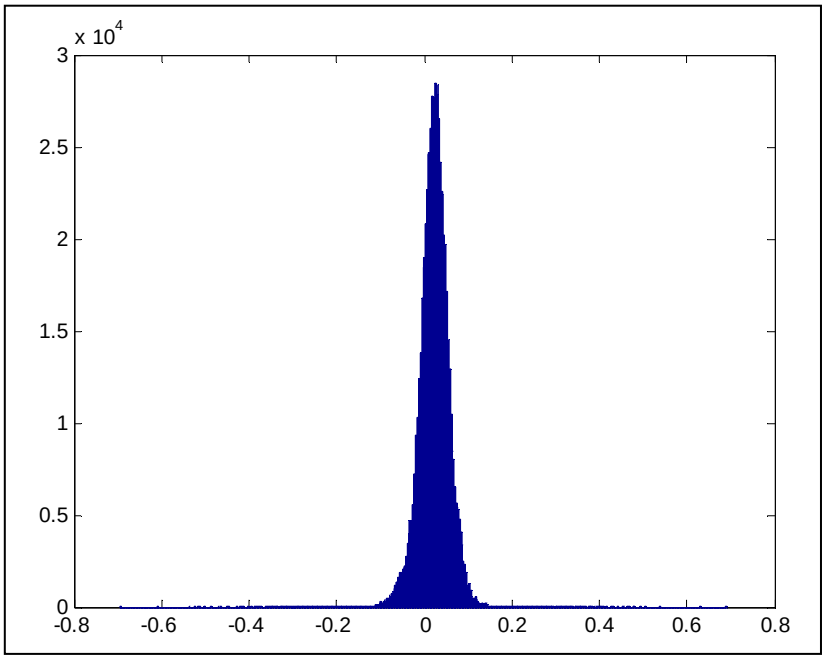

Figure 6.1 Users' Correlations Distribution

Finally, table 6.3 presents the overall performance of each method for the whole experiment. This table simply presents a summary of the results already explained: content-based methods performs better, CF2 goes in second place, despite the amount of failed songs, which is around 10% of the whole data, and the worst one is the random method. Besides, the performances of HA methods are not far away from the performance of CB or CF2. Especially for HA2.1 that present a performance near to the one of CB.

Method[2]	Overall MAE
AV	24.297
RN	42.601
CB	18.031
CF1.1	20.026
HA1.1	20.086
CF1.2	26.143
HA1.2	20.740
CF2.1	18.928
HA2.1	19.087
CF2.2	23.394
HA2.2	19.575

Table 6.3 Overall MAE for every prediction method

[2]AV: Naïve Average method
RN: Naïve Random method
CB: Content-Based.
CF1.1 and CF1.2: Collaborative Filtering with users correlations.
HA1.1: Hybrid Average method between CB and CF1.1.
HA1.2: Hybrid Average method between CB and CF1.2.
CF2.1 and CF2.2: Collaborative Filtering with songs correlations.
HA2.1: Hybrid Average method between CB and CF2.1.
HA2.2: Hybrid Average method between CB and CF2.2.

6.2 Artist Variety

The experiments on artist variety were executed using content-based, collaborative-filtering, and hybrid methods. For this metric there is no point on using the average or random methods. In the first case we will obtain the same rate to all the songs so we will not be able to retrieve top-100 lists. In the second case the idea of artist variety for a random predictor does not make any sense at all. As in the case of recommendation accuracy we will show results in different levels: results per user, and overall results. In the user level we estimated the artist variety for every user in the evaluation test, and for the overall results we compute the average of all the users.

Furthermore, in this second experiment we did not make a distinction between CF1.1 and CF1.2 or CF2.1 and CF2.2. This is mainly because the set of failed songs, whether they receive a prediction of -1 or the average of the user, does not change the top 100 list of any of the collaborative filtering methods. Nevertheless, we did make a distinction between HA1.1, HA1.2, HA2.1 and HA2.2, since the failed songs apparently have a bigger impact on this methods depending whether they receive -1 or the average of the user as a recommendation. We mean that the top-100 lists for HA methods vary depending on which collaborative method was used.

Table 6.4 presents the results for the first 10 users in our evaluation test. As we can see in many of the cases collaborative-filtering recommend songs from a bigger range of artists and in some cases it recommends from more new users than content-based does. From table 6.5 we can observe that in general collaborative-filtering methods produce more artist variety than content-based ones. One interesting point to notice from both tables is that although CF1 recommends more new artists than CB, the ratio between new artists and the total of recommended artist is bigger in CB. This implies that CB recommends a bigger percentage of new artists from the total.

Besides, we can see that HA1.2 outperforms both CB and CF1 and presents a bigger number of total new artists recommended. Even HA1.1 presents a bigger number of new artists than CB or CF1. Furthermore, the amount of new artists obtained by HA2.1 is superior to any other method. From this information we can easily see that hybrid methods present a richer artist variety than pure ones.

Method	Total Artists	New Artists	Ratio
CB	33.970	18.607	0.547
CF1.1	46.950	21.343	0.454
CF1.2	46.950	21.343	0.454
HA1.1	37.833	23.598	0.623
HA1.2	57.588	27.784	0.482
CF2.1	45.813	28.627	0.624
CF2.2	45.637	28.617	0.627
HA2.1	41.676	32.362	0.776
HA2.2	51.745	30.843	0.596

Table 6.5 Overall Average Artist Varieties.

User ID	CB			CF1.1 and CF1.2			HA1.1			HA1.2			CF2.1 and CF2.2			HA2.1			HA2.2		
	T	N	R	T	N	R	T	N	R	T	N	R	T	N	R	T	N	R	T	N	R
11366 65141	49	18	0.367	50	9	0.180	50	23	0.460	64	17	0.265	38	17	0.440	54	39	0.722	54	39	0.722
10158 14370	63	15	0.238	58	9	0.155	64	17	0.265	63	14	0.222	28	11	0.392	56	38	0.678	56	38	0.678
11484 13913	47	28	0.595	43	16	0.372	50	30	0.600	70	28	0.400	42	27	0.642	60	48	0.800	60	48	0.800
12268 75802	65	42	0.646	58	27	0.465	65	42	0.646	60	31	0.516	37	20	0.540	55	39	0.709	55	39	0.709
11362 71492	11	5	0.454	47	11	0.234	13	7	0.538	59	12	0.203	42	26	0.619	20	14	0.699	20	14	0.699
11412 37820	24	13	0.541	38	7	0.184	27	15	0.555	53	7	0.132	26	11	0.423	33	27	0.818	33	27	0.818
11477 39909	59	38	0.644	62	30	0.483	64	44	0.687	60	38	0.633	37	29	0.783	56	49	0.875	56	49	0.875
10277 49191	7	2	0.285	29	4	0.137	7	2	0.285	35	4	0.114	44	22	0.5	15	11	0.733	15	11	0.733
11420 84866	38	19	0.5	65	26	0.400	42	23	0.547	56	22	0.392	37	25	0.675	60	46	0.766	60	46	0.766
11443 10315	33	18	0.545	63	18	0.285	30	16	0.533	55	16	0.291	52	17	0.326	37	33	0.891	37	33	0.891

Table 6.4 Artist Varieties for the first 10 users in the evaluation set

CB: Content-Based.
CF1.1 and CF1.2: Collaborative Filtering with users correlations.
HA1.1: Hybrid Average method between CB and CF1.1.
HA1.2: Hybrid Average method between CB and CF1.2.
CF2.1 and CF2.2: Collaborative Filtering with songs correlations.
HA2.1: Hybrid Average method between CB and CF2.1.
HA2.2: Hybrid Average method between CB and CF2.2.

6.3 Total Coverage

There are many situations in which a specific recommendation method fails to make predictions about certain items. The total coverage metric will give us an idea on how many fails are produced by each of the different prototype. We will present results of the total coverage of the first and second experiment separately. Table 6.6[3] shows the results of the first experiment we can see that in general CB method has really few failed predictions (less than 1%) compared to CF1.1 (30%) or CF2.1 (12%). It is important to notice that for most of the songs that failed with collaborative filtering methods, content-based methods were successful in predicting a value. One of the main reasons for this is that content-based methods only use a single user profile and the information extracted from the content in order to make a prediction. They do not need others users' information to obtain results. Finally, we have to point out the fact that HA method reduced the number of failed song to practically 0 out of the total.

In the specific case of content-based methods, the only situation in which a prediction is not possible it is in the case where the information obtained from the audio file was corrupted or missed. In our collection we have 3 songs with these characteristics which lead to the errors in our method. In the case of collaborative-filtering methods there are more situations that prevent the system for predicting a value for certain items. One situation in which it is not possible to predict items is when a user gave the same rate for all its preferences. In this particular case it is not possible to estimate the correlations with other users since the difference $(S_{xm} - \overline{U}_x)$ in formula 3.2 will be always 0. Another situation where it is not possible to predict items is when certain user present small or no correlation at all with other users. This case is frequent for users with rare preferences. In such cases the estimation of the predicted value it is not possible using formula 3.3.

Fold #	CB	CF1.1	HA1.1	CF1.2	HA1.2	CF2.1	HA2.1	CF.2	HA2.2	Total
1	4	3796	1	0	0	1591	0	0	0	12728
2	1	3715	1	0	0	1609	0	0	0	12728
3	7	3736	0	0	0	1546	0	0	0	12728
4	6	3786	1	0	0	1594	0	0	0	12728
5	3	3841	0	0	0	1595	0	0	0	12728
6	3	3781	1	0	0	1684	0	0	0	12728
7	6	3833	2	0	0	1621	0	0	0	12728
8	5	3810	2	0	0	1612	0	0	0	12728
9	1	3696	0	0	0	1652	0	0	0	12728
10	2	3782	1	0	0	1660	0	0	0	12728
Total	38	37776	9	0	0	16164	0	0	0	127280

Table 6.6 Number of Failed Predictions in the First Experiment

[3] CB: Content-Based.
CF1.1 and CF1.2: Collaborative Filtering with users correlations.
HA1.1: Hybrid Average method between CB and CF1.1.
HA1.2: Hybrid Average method between CB and CF1.2.
CF2.1 and CF2.2: Collaborative Filtering with songs correlations.
HA2.1: Hybrid Average method between CB and CF2.1.
HA2.2: Hybrid Average method between CB and CF2.2.

6.4 Computation times

For both content-based and collaborative-filtering methods the pre-processing step is the one which requires more computational time. In the case of content-based methods the step of estimating the distances between models of songs is the most time consuming. In the case of collaborative-filtering methods the most time consuming step is the one in which correlations between users or songs are estimated. Once the pre-processing steps are done, the time needed for making a prediction is quite small and can be done online easily. Table 6.7 shows the times needed for making 100 predictions with each of the methods. These times indicate that CF1 outperforms content based method in execution time. Although the time presented for HA1 and HA2 are considerable small, we have to take into account that in order to obtain these results first it is necessary to calculate before hand the predictions made by CB, CF1, and CF2. Without pre-computing these predictions the times of HA1 and HA2 might be proportional to the sum of CB and CF times.

# Songs	Time (seconds)				
	CB	CF1	CF2	HA1	HA2
100	30	5	25	< 1	< 1

Table 6.7 Execution times for the different methods.

7 Conclusions and Further work

7.1 Conclusions

The conclusion that we state in this chapter are based on the literature research that we performed during the development of this thesis and also in the results produced by the experiments we conduced with the different methods. First we will discuss about pure methods and their performances. Later in this chapter we will focus on hybrid methods, the advantages we found and their performance. Finally, in the last part of this chapter we will discuss some possible further work that is related to this thesis project and might be subject of study.

From the literature research presented in chapter 3 we learn that collaborative filtering methods are domain independent and they do not need an electronic representation of the songs in order to produce recommendations. We probe these advantages of collaborative filtering methods with the experiments since we didn't use information extracted from the songs to make the predictions. Furthermore the same data and experiments might be applied to any different item such as movies, books, etc. Another of the advantages of collaborative filtering methods that we could prove in this work, is the one related with the variety of the recommendations; from the results obtained n the second experiment we could verify that these methods produce more artist variety that content based methods for instance.

On the other hand, collaborative filtering methods present several disadvantages that are quite easy to confirm. Most of these disadvantages are related with the amount and quality of the data from users' preferences. For instance, in the case of CF1 we obtain low performances due to the low correlation between the users in the system. This situation also leads us to a big amount of items for which it was not possible to make predictions. All theses facts made us conclude that collaborative filtering methods are excessively depend on the external data; and even when there is enough data to produce good results we still can find problems with heterogeneous profiles or profiles of low quality (where the user did not put some effort to state its preferences). In both cases the performance of the system might be low and produce many failed predictions.

Taking into account the results obtained for content based method and in accordance with the advantages presented in chapter 3, we can conclude that this method does not present any troubles when recommending new items. Although this method might present some difficulties for completely new users, this is typical of any recommendation system since in order to make a prediction first it is necessary to learn about the user's preferences. Another important characteristic that was proved on this method is that the features extracted from the audio files have to be estimated only once. There is no need to compute them again if new songs are included in the collection or new users are integrated into the system.

Furthermore, we have proved that content based methods provide more accurate recommendations than other methods and they normally succeed on items that collaborative filtering cannot predict. However, content based method present reduced artist variety. One reason for this is that this method tends to recommend only artists that sound similar to the ones preferred by the user. Another reason is that content based

methods do not take into account other users' opinions. In that sense, several relations between items, which normally are detected by collaborative filtering methods, are lost.

Some final conclusions drawn for pure recommendation methods are related with the fact that most of the computations required to make predictions can be done offline. For instance, estimate similarities among songs (for content based methods) and estimate correlations between users of songs (for collaborative filtering methods) can be done while the system is being developed and when don't have to produce a fast answer immediately. Therefore, these methods can make predictions easily online while the user is waiting for a fast response, in web applications for instance. In the case of content based methods, since they don't need information about other users' preferences, the system might be used even in personal computers were information of a single user is stored.

Moreover, it is important to state that it is hard to improve the performance of content based prototypes. In order to do this it is necessary to use another ways of modeling the songs and re-estimate the similarities among them with a new similarity function. These steps are time consuming and may not produce better results. On the other hand, it is easier to improve the performance of collaborative filtering prototypes since they only need more data about the users' preferences and strong correlations between songs or users. We can conclude that with the correct amount of data collaborative filtering methods can outperform content based ones. However, this amount of data it is not easy to obtain and collaborative methods tend to fail a lot.

Although content based and collaborative filtering methods are capable of producing good results, hybrid recommendation systems help to solve many of the disadvantages of both of them. We drawn this conclusion from the results obtained in our different metrics. For example, our coverage metric proved that if the data about users' preferences is not enough for the collaborative filtering methods to make a prediction then a hybrid system might use information about the content of the songs to make a relevant prediction. Furthermore, when content based method fails on predicting a rich artist variety hybrid methods produce results that are even better than the ones obtained with collaborative filtering. Another interesting point to discuss is that hybrid recommendation systems, especially if they depend on the predictions made by pure methods, can be used easily for making predictions online. As in the case of content-based or collaborative filtering methods, hybrid one need some time to make some strong computations; once that this offline computing is done the time required for making predictions is minimal.

The main conclusion of this work is that content-based and collaborative filtering techniques are complementary ones. Content based techniques help to solve problems when the users' preferences data is inadequate or inexistent and collaborative filtering techniques can obtain relations between songs that are simply impossible to obtain using only the audio files. For this reason we state that it is possible to obtain better results using hybrid recommendation methods instead of pure ones. This conclusion is a direct observation of the results obtained from our experiments and answers our main research question.

7.2 Further Work

Improvement of Content-Based Modeling Methods

Nowadays a lot of research is still going on how to extract truly relevant music features from audio files. Although the use of timbres for modeling musical songs produce good results, they are features that are hard to explain for a common user and they do not take into account important aspects of music such as rhythm or structure. One possible way to improve the results obtained in this thesis is to use another ways of modeling the musical pieces. We strongly believe that including features such as rhythm, or chords in the model of each musical piece will increase the performance of the Content-Based Recommendations. One of the mayor obstacles in extracting other features is related with the fact that we do not posses the audio files. Therefore, in order to use different features might be necessary to purchase the whole collection of audio data.

Improvement of Hybrid Method

As explained in chapter 5, in this thesis we use a simply method to implement our hybrid prototype. Although this method has proven to outperform each pure method in different aspects, we expect that a more complex way of mixing the pure methods will produce even better results. Hence, a further work that can be done is to research into new methods for mixing pure methods. The implementation of the 3-Way Aspect Model might be a initial step in this area, and could help to see if the performance is improved as expected or not.

Experiments with different collections

Another interesting work that could be done ahead is to perform experiments with bigger and more varied collection. For this thesis we use the USPOP collection, which is mainly composed by pop music that it is quite popular in the US. Bigger collections with more genres in it might produce different results. Besides, it is important to see if efficiency and scalability are not a big issue in recommendation systems. Also in this field might be possible to include different ways of evaluating the system. For instance, using human evaluation directly might be interesting and give better insights of the real performance of the systems.

References

[1] **Content-Based Music Audio Recommendation.** Pedro Cano, Markus Koppenberger, Nicolas Wack. ACM international conference on Multimedia. November 6-11, 2005.

[2] **Content-Based Multimedia Information Retrieval: State of the art and Challenges.** Michael S. Lew, Nicu Sebe, Chabane Djeraba, Ramesh Jain. ACM transactions on Multimedia Computing, Communications and Applications. Feb 2006.

[3] **A Highly Robust Audio Fingerprinting System.** Jaap Haitsma, Ton Kalker. International Symposium on Musical Information Retrieval, pp 144-8. 2002.

[4] **Representing Musical Genre: A State of the Art.** Jean-Julien Aucouturier, François Pachet. Journal of New Music Research, Vol. 32, No 1, pp 83-93. 2003.

[5] **Knowledge and Content-based Audio Retrieval using WordNet.** Pedro Cano, Markus Koppenberger, Sylvain Le Groux. 1st International Conference on E-business and Telecommunication Networks. 2004.

[6] **SoundSpotter - A Prototype System for Content-based Audio Retrieval**. Christian Spevak, Emmanuel Favreau. 5th Conference on Digital Audio Effects, Hamburg, Germany. Sep 2002.

[7] **Music Retrieval: A tutorial and Review**. Nicola Orio. Foundations and Trends in Information Retrieval, Vol. 1, Issue 1, pp 1-96. 2006. ISSN 1554-0669

[8] **Content-based playlist generation: exploratory experiments.** Beth Logan. ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France, October 13-17, 2002

[9] **Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences**. Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. ISMIR 2006, 7th International Conference on Music Information Retrieval, pp 296-301, Victoria, Canada, 8-12 October 2006.

[10] **Content-boosted collaborative filtering**. Prem Melville, R. Mooney, and R. Nagarajan. Proceedings of the SIGIR-2001 Workshop on Recommender Systems, New Orleans, LA, September 2001

[11] **Clustering approach for hybrid recommender system**. Qing Li and Byeong Man Kim. 2003 IEEE / WIC International Conference on Web Intelligence, pp 33-38, Halifax, Canada, 13-17 October 2003.

[12] **A review of factors affecting music recommender success**. Alexandra Uitdenbogerd and Ron G. van Schyndel. ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France, October 13-17, 2002

[14] **Music Retrieval based on Melodic Similarity**. Rainer Typke. PhD thesis, Utrecht University, Department of Computer Science, February 2007, ISBN 90-393-4441-8

[15] **ISMIR 2004 audio description contest**. Pedro Cano, Emilia Gomez, Perfecto Herrera. ISMIR 2004, 5th International Conference on Music Information Retrieval, Barcelona, Spain, October 10-14, 2004, Proceedings

[16] http://www.music-ir.org/mirexwiki/index.php/Main_Page

[17] **Content Based music retrieval**. Remco C. Veltkamp, Frans Wiering, and Rainer Typke. Encyclopedia of Multimedia, Borko Furht (Ed.), ISBN: 0-387-24395-X, Springer 2006.

[18] **Evaluation of new Audio Features and their utilization in novel music retrieval application**. Thomas Lidy, Master's Thesis, Vienna University of Technology, December 2006.

[19] **The C-BRAHMS project**. Kjell Lemstrom, Veli Makinen, Anna Pienimaki, Mika Turkia, and Esko Ukkonen. ISMIR 2003 4th International Conference on Music Information Retrieval, pp. 237-238, Baltimore, October 26-30, 2003.

[20] **Signal processing methods for the automatic transcription of music**. Anssi Kapluri. Ph.D. thesis, Tampere University of Technology, Finland, April 2004.

[21] **The Shazam music recognition service**. Avery Wang. Communications of the ACM, Vol. 49, No. 8, pp 44-48, 2006

[22] **Morphological sound description: computational model and usability evaluation**. Julien Ricard and Perfecto Herrera. Proceedings of the 116th Convention of the Audio Engineering Society, 2004.

[23] **A study on Content-based classification and retrieval of audio database**. Mingchun Liu, Chunru Wan. Proceedings of International Database Engineering & Applications Symposium, IDEAS 2001, July 16-18, 2001, Grenoble, France.

[24] **Music recommendation from song sets**. Beth Logan. ISMIR 2004, 5th International Conference on Music Information Retrieval, Barcelona, Spain, October 10-14, 2004, Proceedings.

[25] **Pitch histograms in audio and symbolic MIR**. George Tzanetakis. ISMIR 2002, Proceedings 3rd International Conference on Music Information Retrieval, pp 31-38 Paris, France, October 13-17, 2002

[26] **Tonal description of polyphonic audio for music content processing**. Emilia Gomez. Informs Journal on Computing, Vol. 18, No. 3, pp. 294-304, Summer 2006.

[27] **Looking for new, not known music only: Music retrieval by melody style**. Fang-Fei Kuo, Man-Kwan Shan. ACM/IEEE Joint Conference on Digital Libraries, JCDL 2004, pp 243-251, Tucson, AZ, USA, June 7-11, 2004, Proceedings.

[28] **Estimating the tonality of polyphonic audio files: Cognitive versus machine learning modeling strategies**. Emilia Gomez, Perfecto Herrera. ISMIR 2004, 5th International Conference on Music Information Retrieval, Barcelona, Spain, October 10-14, 2004, Proceedings

[29] **Mel Frequency Cepstral Coefficients for music modeling**. Beth Logan. ISMIR 2000, 1st International Symposium on Music Information Retrieval, Plymouth, Massachusetts, USA, October 23-25, 2000, Proceedings.

[30] **A music similarity function based on signal analysis**. Beth Logan, Ariel Salomon. Proceedings of the 2001 IEEE International Conference on Multimedia and Expo, ICME 2001, August 22-25, 2001, Tokyo, Japan

[31] **A music recommender based on audio features**. Qing Li, Byeong Man, Dong Hai. SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 532-533, Sheffield, UK, July 25-29, 2004

[32] **Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments**. Alexandrin Popescul, Lyle Ungar, David Pennock. UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, pp 437-444, University of Washington, Seattle, Washington, USA, August 2-5, 2001

[33] **Improving content-based similarity measures by training a collaborative model**. Richard Stenzel, Thomas Kamps. ISMIR 2005, 6th International Conference on Music Information Retrieval, pp 264-271, London, UK, 11-15 September 2005, Proceedings

[34] **Supporting people in finding information. Hybrid recommender systems and goal-based structuring**. Mark Van Setten. Telematica Instituut Fundamental Research Series, vol. 016. Enschede, the Netherlands: Telematica Instituut, 2005 ISBN 90-75176-89-9

[35] **A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures.** A. Berenzweig, B. Logan, D. P. W. Ellis, and B. Whitman. In Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR'03), Washington, D.C., USA, 2003

[36] **Modularity of music processing.** Isabelle Peretz and Max Coltheart. Nature Neuroscience 6, pp 688 – 691. 25 June 2003.

[37] **Survey of music information needs, uses, and seeking behaviors**. J. H. Lee and J. S. Downie.: Preliminary findings. In ISMIR Proceedings, pages 441– 446, 2004.

[38] **Towards Evaluation Techniques for Music Similarity.** B. Logan, D. Ellis, A. Berenzweig. White paper at the Workshop on the Evaluation of Music Information Retrieval Systems at SIGIR-03. Toronto, August 2003.

[39] **The Quest for Ground Truth in Musical Artist Similarity.** D. Ellis, B. Whitman, A. Berenzweig, S. Lawrence. Proceedings of ISMIR-02. Paris, France. October 2002.

[40] **The Music Information Retrieval Evaluation eXchange (MIREX). Conference Report.** J. Stephen Downie. D-Lib Magazine. Vol. 12, No 12. December 2006.

[41] **Audio Music Similarity MIREX 2006 web site.** http://www.music-ir.org/mirex2006/index.php/Main_Page

[42] **Official Description of the Evalutron 6000 System and Protocol.** http://www.music-ir.org/mirex2006/index.php/Evalutron6000_Issues

[43] **All Music Guide web site.** www.allmusic.com

[44] **Yahoo LaunchCast Music web site.** www.music.yahoo.com

[45] **Music information retrieval (Chapter 7).** Downie, J. Stephen. In Annual Review of Information Science and Technology 37, pp 295-340. Medford, New Jersey, USA. 2003

[46] **A Review of Factors Affecting Music Recommender Success.** Alexandra L. Uitdenbogerd and Ron G. van Schyndel. ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France, October 13-17, 2002, Proceedings.

[47] **Social Information Filtering: Algorithms for automating "Word of mouth"**. Upendra Shardanand, Pattie Maes. Proceedings of CHI'95 Conference on Human Factors in Computing Systems, volume 1, pp 210-217. ACM Press, 1995.

[48] **Web Services for Music Information Retrieval**. Mark Zadel, Ichiro Fujinaga. Proceedings of 5th International Conference on Music Information Retrieval. Barcelona, Spain, 2004

[49] **Web-collaborative filtering: recommending music by crawling the Web**. William W. Cohen, Wei Fan. Computer Networks, volume 33, number 1-6, pp 685-698. Amsterdam, Netherlands, 1999.

[50] **Social Information Filtering for Music Recommendation**. Upendra Shardanand. S.M. Thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology, 1994

[51] **Computational Models of Music Similarity and their Application to Music Information Retrieval**. Elias Pampalk. Doctoral Thesis, Vienna University of Technology, Austria, March 2006

[52] **Ensemble Learning for Hybrid Music Recommendation**. Marco Tiemann, Steffen Pauws, Vignoli, Fabio. Proceedings of 8th International Conference on Music Information Retrieval, pp 179-180. Vienna, Austria, 2007.

[53] **Similarity Based on Rating Data.** Malcolm Slaney, William White. Proceedings of 8th International Conference on Music Information Retrieval, pp 479-484. Vienna, Austria, 2007

[54] **Evaluation of Distance Measures between Gaussian Mixture Models of MFCCS.** Jesper Højvang Jensen, Daniel P. W. Ellis, Mads G. Christensen, Søren Holdt Jensen. Proceedings of 8th International Conference on Music Information Retrieval, pp 107-108. Vienna, Austria, 2007

[55] **Improving Efficiency and Scalability of Model-Based Music Recommender System Based on Incremental Training.** Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Proceedings of 8th International Conference on Music Information Retrieval (ISMIR 2007), pp.89-94. Vienna, Austria, 2007

Appendix A – Database Schema

The database schema used to store all the information necessary to develop the systems is composed by 9 main tables and several temporal ones where the results of the recommendations were stored. The nine tables can be grouped according to their purpose and which of the recommendation methods use them. The first group contains general information tables such as songs names, authors, rates, etc. The second group contains the tables that are used by the content-based recommendation systems. Finally, the third group is formed by the tables used by the collaborative-filtering recommendation prototype. In the next paragraphs we will explain in more detail each of these groups, its tables and relations.

The general purpose group it is formed by tables that contain metadata information about the songs, the users and the rates. These tables are used by all the prototypes at certain point and represent the result of the recollection information step in the recommendation process, see chapter 2 for more details. The main tables of this group are *songs_rates* and *songs*. The first one stores all the information about the rates of all the users. Since this table stores near to 3 million rates it is necessary to index it for fast access. This table uses a BTree clustered index on the UserID and SongID fields. Furthermore, the table *songs* stores general information about the songs. For the purpose of the experiments we stored information such as the song title, the song's album, and song's authors.

The second group of tables is used by the content-based prototype and they store information about the audio files and their distances. The tables that form this group are *sources_relations* and *distance*. Table *distance* stores all the Kullback-Leibler divergences between all the songs. Since this table also stores a big amount of data, more than 64 millions of rows, it is necessary to index it for efficient retrieval. The index used in this table is also a BTree clustered index in the fields *RowID* and *ColumnID*. On the other hand, the table *sources_relations* is a special table that it is used to correlate the audio files with the information downloaded from the Yahoo web page. This table is quite important for the content-based prototype since it relates the physical audio files with the rates given by the users.

Finally, the third group of tables is made up by the tables which store the information that the collaborative-filtering prototype uses to make recommendations. These tables are *users_correlations* and *songs_correlations*. Both tables store the correlations among users and songs. Table *users_correlations* is used the first collaborative filtering method (CF1) while *songs_correlations* is used by the second one (CF2). Since both tables store a big amount of rows, we use BTree clustered indexes in both tables for fast retrieval.

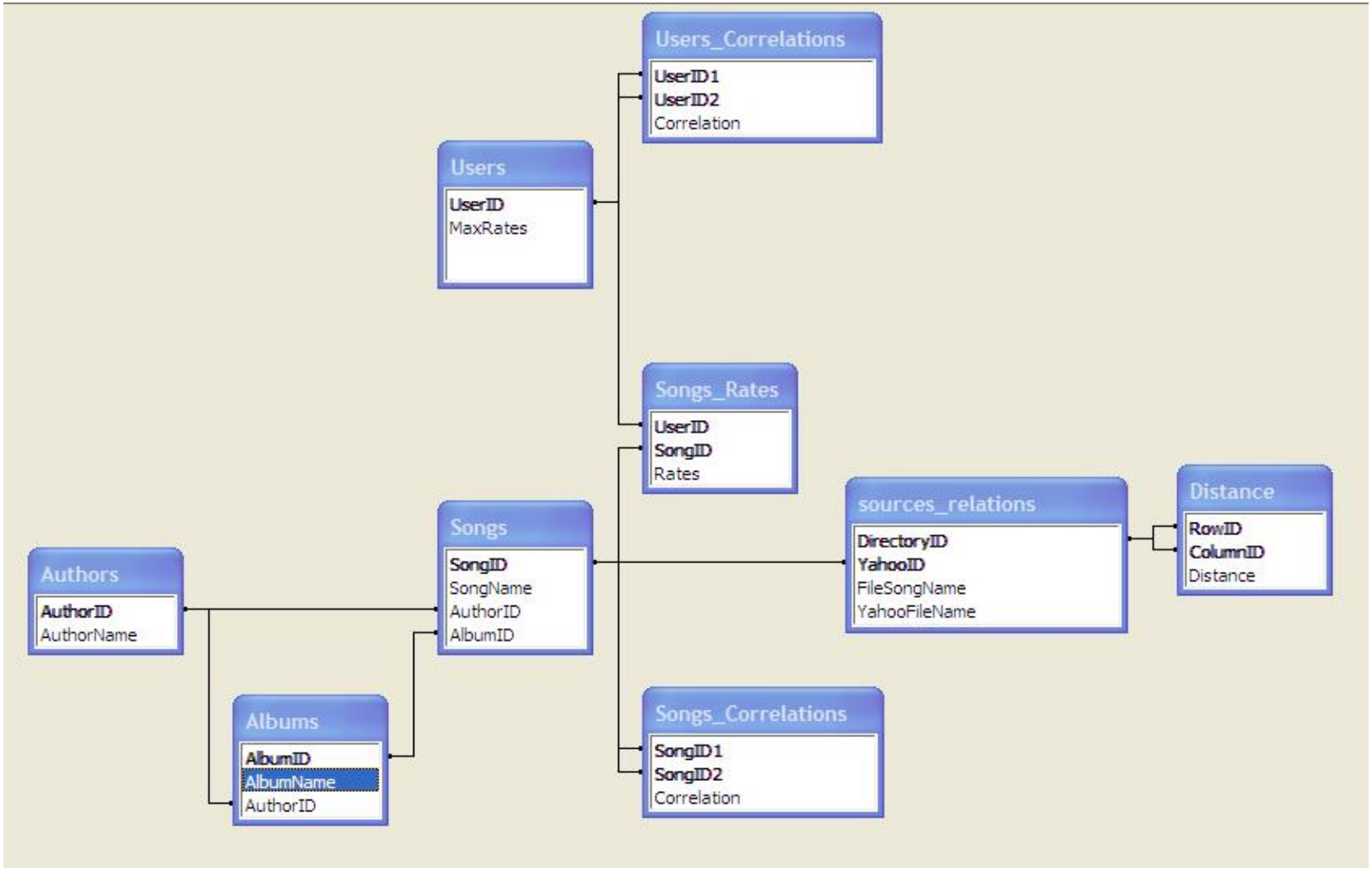

Figure A.1 Database Tables and Relationships