MONOGRAFIES DE L'INSTITUT D´INVESTIGACIÓ EN INTEL·LIGÈNCIA ARTIFICIAL

Poolcasting:
an intelligent technique to customise music programmes for their audience

Claudio Baccigalupo

Consell Superior d´Investigacions Científiques

MONOGRAFIES DE L'INSTITUT D'INVESTIGACIÓ
EN INTEL·LIGÈNCIA ARTIFICIAL
Number 37

Institut d'Investigació
en Intel·ligència Artificial

Consell Superior
d'Investigacions Científiques

Poolcasting: an intelligent technique to customise music programmes for their audience

Claudio Baccigalupo

Foreword by Enric Plaza

2010 Consell Superior d'Investigacions Científiques
Institut d'Investigació en Intel·ligència Artificial
Bellaterra, Catalonia, Spain.

Series Editor
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Foreword by
Enric Plaza
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Volume Author
Claudio Baccigalupo
Institut d'Investigació en Intel·ligència Artificial
Consell Superior d'Investigacions Científiques

Institut d'Investigació
en Intel·ligència Artificial

Consell Superior
d'Investigacions Científiques

NIPO: 472-10-155-6
ISBN: 978-84-00-09149-1
Dip. Legal: B.46003-2010

Ordering Information: Text orders should be addressed to the Library of the IIIA, Institut d'Investigació en Intel·ligència Artificial, Campus de la Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, Spain.

Contents

List of Figures

Foreword

The research work presented in this monograph inaugurates a research line of high future potential: applying Artificial Intelligence techniques, and singularly Case-based Reasoning, to data in the World Wide Web embodying information about common activities generated by large number of people. Claudio Baccigalupo has developed Poolcasting, a system that exemplifies these approach in the music domain.

Analysing the activity generated by large number of people in the musical domain, Claudio Baccigalupo shows that innovative, intelligent services can be provided for the users in that same domain. Specifically, the Poolcasting system an audience-customized music programming service in real time. Interestingly, Poolcasting can be seen as a system that generalises the principles of Case-based Reasoning, in which Poolcasting follows the core idea of CBR, namely reasoning and learning from experience, while "experiences" are not directly expressed as "cases." This contribution was acknowledged by the Best Application Paper Award of the 2007 International Conference on Case-based Reasoning.

Finally, the ideas developed in the course of Claudio Baccigalupo's Ph.D. have had the effect of showing how CBR can be applied in new ways to the data generated by the activity of large number of people in the World Wide Web. Two workshops have been organised along this line, with the title "Web CBR: Reasoning from Experiences in the Web." This monograph will allow the readers to dive into these new ideas with the specific application to music giving a clear and rich context in which to understand how the interplay of ideas and concrete developments is the core of Artificial Intelligence as a research program.
Bellaterra, December 2010

Enric Plaza
Research Professor, IIIA-CSIC

Abstract

Poolcasting is an intelligent technique to customise musical sequences for groups of listeners. Poolcasting acts like a disc jockey, determining and delivering songs that satisfy its audience. Satisfying an entire audience is not an easy task, especially when members of the group have heterogeneous preferences and can join and leave the group at different times. The approach of poolcasting consists in selecting songs iteratively, in real time, favouring those members who are less satisfied by the previous songs played.

Poolcasting additionally ensures that the played sequence does not repeat the same songs or artists closely and that pairs of consecutive songs 'flow' well one after the other, in a musical sense. Good disc jockeys know from expertise which songs sound well in sequence; poolcasting obtains this knowledge from the analysis of playlists shared on the Web. The more two songs occur closely in playlists, the more poolcasting considers two songs as associated, in accordance with the human experiences expressed through playlists. Combining this knowledge and the music profiles of the listeners, poolcasting autonomously generates sequences that are varied, musically smooth and fairly adapted for a particular audience.

A natural application for poolcasting is automating radio programmes. Many online radios broadcast on each channel a random sequence of songs that is not affected by who is listening. Applying poolcasting can improve radio programmes, playing on each channel a varied, smooth and group-customised musical sequence. The integration of poolcasting into a Web radio has resulted in an innovative system called Poolcasting Web radio. Tens of people have connected to this online radio during one year providing first-hand evaluation of its social features. A set of experiments have been executed to evaluate how much the size of the group and its musical homogeneity affect the performance of the poolcasting technique.

Acknowledgements

This work is dedicated to everyone who made my research possible and my life pleasurable during the last four years.

To Enric Plaza for the tips, corrections, and constant work. To MusicStrands for financing the scholarship that paid for my studies. To the founders of the IIIA for creating this unrivalled Research Institute. To Andrea Giovannucci for making me come to Spain in the first place. To Luca Merlino for sharing many lunches and ideas. To Justin Donaldson for making me addicted to R. To Elias Pampalk for sharing my work with Last.fm. To Neal Lathia and Daniele Quercia for inviting me at the University College of London. To Amélie Anglade for inviting me at the Queen's Mary University of London. To Lorenzo Bernacchioni and Manu Atencia for sharing their personal music collections. To Jim Shur for packing up the data set of playlists. To Tito Cruz for the supreme technical support. To David Baelde for creating liquidsoap. To Aaron Wolfe for creating Spogbiper. To David Heinemeier Hansson for creating Ruby on Rails. To Michael Casey, Derek Bridge, Peyman Faratin, Francisco Martín, Barry Smyth for the encouragement. To Enrico and Marita, my parents, for the everlasting support. And to Pau for relentlessly listening to me while reading out loud the thesis uncountable times, though this was not mentioned in our marriage contract. Thanks.

Chapter 1

Introduction

I ask in all honesty, what would life be?
Without a song or a dance what are we?

ABBA, 1983

1.1 Reusing past experience

Internet is a rich and outspread medium which allows anyone to easily contribute to its content. People share their pictures, videos, songs, texts and, very often, their *personal experiences*, reporting which songs they have played, which friends they have met, which places they have visited, and so on. Sharing personal experience online has become a common activity, which explains the success of Web communities such as Facebook and Twitter offering this kind of service.

Experience data from the Web grants a valuable overview of the social impact of a given content. Browsing the opinions of previous travellers in online forums, for instance, is often more beneficial to decide which hotel to book than the information provided by a guide book. The same occurs with books, videos, music, technical problems, recipes, advices: collecting experiences of others from the Web can offer the knowledge required to perform several tasks.

This dissertation presents poolcasting, an Artificial Intelligence technique that takes advantage of the vast amount of experiential data shared on the Internet to automatically solve a specific task. The task addressed is to customise a sequence of songs for the preferences of a given audience.

There are several situations where people listen to music in groups (discos, radio channels, home-parties) and a professional disc jockey (DJ) is appointed to select the best songs to play for the current audience. Poolcasting is designed to act 'like a good DJ', selecting *automatically* which songs to play from a repository of available music.

While a DJ combines personal expertise and human senses to 'feel' which music is adapt for a given audience, the innovation of poolcasting is to obtain this knowledge from the World Wide Web. Internet makes available a wide repository

of human experiences related to the domain of music: which songs people have played in the past, in which order, how they were rated, commented, forwarded, shared. Poolcasting collects and interprets this kind of human experiences to obtain the knowledge required to deliver smooth and group-customised musical sequences.

The core idea of poolcasting is to solve a task reusing past experiences. This idea characterises a whole family of Artificial Intelligence approaches known as **Case-Based Reasoning** (CBR). CBR systems typically store past experiences as cases, then reuse previous cases to propose solutions for new tasks. A CBR system, for instance, would determine the best cure for a patient by first retrieving similar past problems (patients with the same profiles and symptoms) from a case base, then adapting their solutions (applied cures and effects) to the current case.

Poolcasting represents a reinterpretation of the classical Case-Based Reasoning approach. To solve a task, previous knowledge is reused, but not structured as (problem → solution) pairs. The task to be solved is not to find *one* solution for a given problem, but to iteratively build a good *sequence* of songs that satisfies certain properties in the long run. Poolcasting demonstrates how CBR can make use of the experience of *multiple* users to solve a task that is *social* in nature: to customise content for a group of people.

1.2 A Web of musical data

A focus of this work is on the area of **Web data mining**, describing a technique to collect, analyse, filter and interpret experiential data from the Web to extract valuable knowledge to solve a specific task.

Internet is expanding towards a social medium as more user-generated content is becoming available offering valuable insights about human experiences. This is particularly true in the domain of music.

Music has, on one side, a mathematical nature that has been investigated for centuries and, on the other side, a social nature that has comparatively received less attention. Yet, people commonly influence one another with respect to the music they like and often spend time looking for the 'right' group of people whom to listen music with. On the Web, music lovers can gather in forums to talk about their favourite artists, form virtual fan clubs to show their support, write entries in their blogs and comments in MySpace pages to explain their interests, discover songs that other friends recently played, forward their preferred music to listeners around the world. Music is a universal language, and people of any age, provenience or preference can report their musical experiences of the Web.

One particular type of musical experience data that is widely available on the Internet is playlists. Playlists are sequences of music titles compiled to be played in a specific order. Playlists are useful since they allow people with large music libraries to organise songs in small ordered sequences.

Each playlist can reflect a particular mood or emotion or be shaped by a specific purpose: working, running, cooking, going to sleep. Even ignoring the

motivation that drove someone to put songs in a certain sequence, a playlist indicates that certain songs have been *experienced* together.

Many music-related Web communities invite their members to share personal playlists online, making others aware of songs that are, in their experience, meant to be played together. Millions of playlists are therefore publicly available on the Internet. Poolcasting collects and analyses a large set of these playlists to reveal songs and artists that are correlated according to the people. If two songs or artists co-occur often and closely in multiple playlists, poolcasting understands that those songs and artists share some affinity and it makes sense to play them together. In this way, poolcasting is able to generate sequences of songs that go well together one after the other.

The innovation of this approach is that poolcasting can identify songs that are associated according to the *people*. Other features could be analysed to uncover songs that are similar (e.g., similar lyrics, similar chords, similar acoustic features), but a content-based analysis would not bring a comprehensive overview of songs that people tend to associate. The experiences concealed in playlists, on the other hand, have the power to reveal songs that people associate either for acoustic, social or cultural reasons.

1.3 Listening to music in a group

Another focus of this thesis is related to the research in the area of **social choice**. The purpose of poolcasting is to customise music for a *group of listeners* (an audience), and this activity implies looking for an acceptable compromise among the musical preferences of the entire audience.

The most appreciated disc jockeys are those that can build 'good' musical sequences for every type of audience, making all the participants move to the dance floor during an event, and not just a few people. Similarly, the goal of poolcasting is to deliver music that can possibly satisfy the entire group.

If a group were made of people with identical musical tastes, it would be easy to find songs that *everyone* likes. This scenario, though, is quite rare: most groups show different individual preferences for different songs, and what someone likes, someone else dislikes.

Under these conditions, the approach of poolcasting is to build a musical sequence that fairly satisfies all the listeners *in the long run*. At times, people may be exposed to songs they do not particularly like but, after a certain while, everyone will be satisfied by the overall music played.

The way in which poolcasting tackles this social issue is by introducing a new preference aggregation method that assigns different importance to different members according to how much satisfied they are towards the music played so far. This satisfaction-weighted aggregation method is designed to select, at each moment, songs that are most liked by the least satisfied listeners so that, in the long term, a balance can be reached in the entire audience.

1.4 A social radio experience

While the first part of the thesis describes the poolcasting technique, the second part introduces an application developed on top of this technique. The application, called Poolcasting Web radio, provides an innovative online radio service that streams music channels on the Internet, where songs are selected in real time according to the preferences of the connected listeners.

In the real world, people are used to listen to music in groups. In bars, cars, elevators, open offices, people share musical experiences and implicitly accept a social contract by which their musical interests only partially influence the music played, which is intended to satisfy the group as a whole. This kind of social experience does not occur on the Internet.

When listening to music on the Web, people typically choose between online radios or digital music services. Online radios are equivalent to terrestrial radios; they just stream over the net rather than over the air and have a much larger number of channels. They are not social, since connecting to a channel does not bring any information about who else is listening, nor allows like-minded listeners to know each other. Moreover, online radios are not personalised: they broadcast either random or pre-programmed musical sequences that are not influenced by the listeners.

The alternative to online radios is provided by digital music services such as Last.fm and Pandora which stream personalised music channels. These services compile a music profile of each member and select songs according to the preferences of each individual. Again, they lack of a social component since streams can only be listened by one person at the time and cannot be shared.

The fact that the Web is becoming more and more a *social* medium and that online radios and digital music services are only targeted to *individuals* is the motivation that led to the development of Poolcasting Web radio, which offers an online *social radio* experience.

Poolcasting Web radio is not meant to help people share music *files*, but to share *musical experiences*, allowing displaced persons to listen to the same music at the same time, to exchange their interests, to contribute together to the sequence of music played and to be able to discover new music that others like.

1.5 Structure of the thesis

Each chapter of the thesis touches a particular research field; for this reason the state of the art related to each area is reported separately at the beginning of each chapter.

Chapter 2 is dedicated to the experience Web and explains how Internet can be mined for data related to musical experiences in the form of playlists, and how playlists can be analysed to learn which songs and artists are more associated according to the 'wisdom of the crowd'. Previous work about co-occurrence analysis and musical associations is also presented.

Chapter 3 explains how to generate individual music profiles from listening behaviour data. Most digital music players store data about individual listening history (which songs a person has played and rated) and many users agree to share these data on the Web to make others aware of their musical experience. This chapter first introduces previous work about user modelling and then describes a technique to analyse listening behaviour data to infer a model of the musical preferences of each listener.

Chapter 4 illustrates the core of the dissertation: the poolcasting technique. Given a group of listeners and a repository of songs, poolcasting employs the techniques of Chap. 2 and 3 to collect the knowledge required to determine a sequence of songs adapted to a given audience. Poolcasting is characterised by an iterated CBR process that first *retrieves* good candidate songs to be added to the sequence, next *reuses* the acquired knowledge to identify the best candidate, then *revises* the knowledge to customise the solution for the current problem. The chapter reviews state of the art about Case-Based Reasoning and group-adaptive systems and also introduces the satisfaction-weighted aggregation method employed by poolcasting to achieve fairness in the long run.

Chapter 5 first reviews previous work about group-adaptive music systems and Internet radios and then describes Poolcasting Web Radio, the innovative application developed to provide an online social radio service. Each channel of the radio is driven by a process that selects in real time which songs to broadcast according to the preferences of the listeners in the same channel at each moment. The radio offers the chance to share music and listening experiences to listeners located around the world.

Chapter 6 reports the evaluation of the presented work, showing the degree in which musical sequences can be customised for a given audience while maintaining a certain musical continuity from song to song. Different scenarios are compared, measuring the performance of poolcasting with groups of different size and degree of homogeneity in terms of musical interests.

Chapter 7 summarises the contributions of this research work and suggests ways to extend poolcasting to domains other than music, in order to deliver customised sequences of movies, news items or TV shows to groups of people.

Chapter 2

Musical associations from a Web of experiences

DJ, don't you dare to leave
I tried to figure out what others did

Dover, 1999

2.1 The experience Web

The World Wide Web keeps one billion persons connected [comScore, Inc., 2009] and has just turned eighteen [Berners-Lee, 1991]. In its come of age, the Web has grown not just in quantity, but most importantly in quality. Old-fashioned media have spent years questioning the reputation of Internet as a trustworthy source [Johnson and Kaye, 2004] and have been outwit. Internet offers worldwide information with a swiftness and a level of granularity that cannot be found in books, newspapers, radios or televisions while preserving the same accuracy [Giles, 2005].

The power for the Internet to evolve from a limited and disregarded to a rich and outspread medium comes from enabling anyone to easily contribute to its content. While Web publishing in the 1990s was cumbersome and expensive, nowadays a few clicks are enough to share content on the Web, with publishing services such as YouTube (video), MySpace (music) and Flickr (images) having become of public domain.

The most remarkable effect of enabling people to easily share content has been to have more people create new content. Most blogs' writers, for instance, never had a personal diary on paper, while millions of pictures, videos, animations, stories are created on purpose to be broadcast online. Internet provides content that would simply be unavailable or inexistent otherwise.

2.1.1 A repository of personal experiences

A particular type of content that has recently gained ground in the Web is **experience data**. This refers to data describing what people have been doing: where they have been, which songs they have played, which friends they have met, and so on. Most people enjoy sharing this personal information, which explains the success of Web services such as Facebook and Twitter, where no original content is actually created. What is shared are *personal experiences*, in such a quantity that cannot be found in any other place.

Experience data from the Web grants a valuable overview of the social impact of a given content. An example is provided by Panic! At The Disco, a Rock band that gained online fame thanks to the 'buzz' generated in the music communities MySpace and PureVolume. While unsigned bands typically have a reduced group of listeners in these Web sites, Panic! At The Disco singularly broke this rule, with more people playing their songs week after week and eventually reaching number one in MySpace's indie chart, which brought them to sign a contract and sell almost four million copies with their first album.

The lesson learnt is that knowing what many people are doing online with music is very valuable. Music labels are now aware of the importance of online experience data and more unsigned artists are reported to have been offered contracts because they had many online listeners or many 'friends' (MySpace jargon for 'supporters') shared with other famous musicians. The strategical importance of collecting musical experience data from the Web is also confirmed by MySpace and Last.fm — two of the main music-driven online communities — being recently acquired by media corporations News Corp. and CBS Interactive for 580 million dollars and 280 million dollars respectively [News Corporation, 2005; CBS Corporation, 2007].

2.1.2 Collecting music-related experiences

Extracting music-related experience data from the Web is not easy task. People have a hand of choices to narrate their 'musical experiences' online: writing a post about a new song in a blog, reporting part of the lyrics in a forum, watching the video in YouTube, playing the audio in Last.fm, becoming friend of the band in MySpace: all these behaviours denote an interest for a song, yet they are all stored in different sites and cannot be easily aggregated.

This is particularly true for experiences described in a *textual* format. Blog posts, for instance, can report both positive and negative music experiences (an awesome concert, a disappointing album); to really understand what is said about music, one would have to read the whole text, filter out typo errors, identify where the music is referred to and categorise the experience.

A human could perform this process only on a small number of blog posts, certainly not on all the posts that bloggers generate daily. On the other hand, an automatic system would have it hard to understand what a blog post is about, given the infinite quantity of languages, styles and locutions that can describe musical experiences.

Luckily, there are musical experiences that are shared on the Web in non-textual formats and that, as such, do not require an *interpretation* since they clearly imply which music a person has experienced and how. The main format to describe songs that have been played and enjoyed is that of playlists.

2.1.3 The value of playlists

Playlists are ordered sequences of music titles. Many persons use playlists in their digital players to specify which songs they would like to hear for a given occasion, for instance reading, programming, driving, working, waking up, going to sleep [Vignoli, 2004]. People also create playlists that reflect a particular mood or emotion, that tell a story or are meant to be shared with friends [Cunningham et al., 2004]. Playlists can be thought of as a "collective memory" [van Dijck, 2006] of the way in which people organise their music.

Playlists are very valuable to automatically extract experience data about music from the Web because they are are less prone to interpretation errors. Each playlist has a clear structure: a sequence of songs that a person compiled to be played together. Even ignoring the *motivation* that drove someone to put songs in a certain sequence, the same effort of compiling and publishing a playlist indicates a preference for those songs.

The singularity of playlists, compared to other experience data (blog posts, album reviews, concert reports), is that playlists intrinsically bind *multiple* songs and artists together. Someone compiling and sharing a playlist is not expressing an interest for a single song, but for a particular sequence of songs meant to be played in a specific order. For this reason, playlists not only offer an overview of which songs or artists are more or less played, but also of which songs or artists people tend to *associate*, to play together.

2.1.4 From playlists to musical associations

When people compile playlists, they include songs that are meant to be played together for some purpose. If two songs or artists appear in many playlists by many different authors, then it makes sense to assess that those songs or artists are *associated*.

The purpose of this chapter is to define a measure of **musical association** to estimate how much two songs or artists 'go well together' based on their playlist co-occurrences. The approach consists in retrieving playlists from Internet and analysing the co-occurrences of songs and artists in such playlists to determine how much they go well together.

The hypothesis is that if any association (cultural, social, or acoustic) exists between two songs or artists, this can be automatically revealed observing how a multitude of people have organised their music in playlists shared on the Web. Playlists hide valuable personal experiences; this chapter shows how these experiences can be extracted and musical associations uncovered.

2.2 Previous work

The approach described in this chapter explains how to reuse experience data included in playlists to assess a measure of musical association between songs and artists. This section reviews previous work addressed to estimate musical associations among musical objects, with a particular emphasis on methods that exploit data retrieved from the Web.

2.2.1 Co-occurrence analysis

Some researchers assume that playlists contain "similar music" [Logan et al., 2003] and that, if two songs are contiguously one after the other in a playlist, it somehow makes sense to play them in this order [Andric and Haus, 2006]. Although this assumption may not hold for one playlist, applying this hypothesis to a large data set of playlists implies that associated songs can be discovered when they co-occur in many playlists.

Identifying relevant *co-occurrences* of items in a series of events is a common issue in unsupervised learning [Hofmann and Puzicha, 1998]. Co-occurrence analysis has been applied not only to identify associated songs from a set of playlists, but also to find maximal frequent sequences in text [Ahonen-Myka, 1999], to discover interesting episodes in the alarm flow of telecommunications networks [Mannila et al., 1997], to gather knowledge about DNA sequences [Wang, 1997] or to learn Web navigation models [Nakagawa and Mobasher, 2003].

The main benefit of co-occurrence analysis is that it allows to state whether two resources are associated or not, not because of some intrinsic content, but because of they way objects are organised. In the case of music, assessing that two songs are associated because they co-occur in many playlists is a way to reveal the "wisdom of the crowd" [Surowiecki, 2004], the way in which songs are most associated by people in their activities.

The basic idea of co-occurrence analysis is that the higher the *number* of playlists where two songs co-occur, the higher their association. As Andric and Haus, 2006 [Andric and Haus, 2006] first noticed, the *order* in which songs appear in playlists is another important property that should be tracked to determine their association. Ragno et al., 2005 [Ragno et al., 2005] suggested that the *distance* at which songs co-occur in a playlist is also relevant: songs appearing contiguously in a playlist are more enlaced that songs divided by a certain time gap. Pachet et al., 2001 [Pachet et al., 2001] indicated how the *popularity* of each song is also fundamental in the co-occurrence analysis since popular songs tend to occur in playlists with many other songs, independently of any real musical association.

Co-occurrence analysis can be performed at different level of *granularity*, focusing either on the co-occurrences of songs, of albums, or of artists. Hauver and French, 2001 [Hauver and French, 2001] suggested that, since songs performed by the same artist are often similar, working at the level of artists can be useful, especially when the data set is small or sparse.

2.2.2 Playlists

To analyse co-occurrences of songs in a large set of playlists, first many playlists compiled by different persons need be retrieved from the Web. Playlists can be found in millions throughout Web pages of different countries and cultures. Many authors publish their personal playlists online, either directly from their music players (e.g., Apple iTunes, Spotify) or inside online communities, the most popular being Imeem, Playlist.com, FIQL, Last.fm, MusicStrands, SonicSwap, Art Of The Mix, Plurn and UpTo11.

Although there is no unique way to distribute playlists online, the most common representation is the XML Shareable Playlist Format (XSPF) [Gonze et al., 2006] that specifies, for each song in the playlist, its title, artist name, album name, song length and location.

Sometimes a song can appear in different databases with different titles, and this can make it difficult to compare playlists shared among different communities. For instance the song 'The Bitter End' (Placebo) is represented as 'The Bitter End' (Placebo) in Last.fm, as 'Bitter End, The' (Placebo (Pop)) in MusicStrands, and as 'Placebo-Bitter End' () in Plurn. The main reason for this misalignment is that, although the music industry has defined an ISO standard music title reference — the ISRC code [ISRC Agency, 2003] — this is usually not used in existing information database, as pointed out by Pachet et al., 2001 [Pachet et al., 2001].

Possible solutions to univocally identify music titles are to either use audio fingerprinting [Cano et al., 2005], music ontology matching [Raimond and Sandler, 2007] or to query large online public music databases such as Musicbrainz.

2.2.3 Other Web-based musical experience

Playlists are not the only type of online data that can be analysed to assess associations between songs and artists.

Cohen and Fan, 2000 [Cohen and Fan, 2000] first suggested that co-occurrences of musical artists in Web *search results* reflect the way in which people associate artists in the music domain. On the same line, Whitman and Lawrence, 2002 [Whitman and Lawrence, 2002] described a method which first queries search engines for pages related to an artist, next extracts natural language features from these pages to produce a summary description of the artist, and finally uses this description to compute similarity between artists. Zadel and Fujinaga, 2004 [Zadel and Fujinaga, 2004] proposed to combine Web services from Amazon and Google to generate pools of potentially related artists and use Web search result counts to assess their actual relatedness. Schedl et al., 2006 [Schedl et al., 2006] also employed Web services to search for combinations of artists names with keywords like 'music', 'genre', 'style'.

The main limitation of these methods, as observed by Levy and Sandler, 2007 [Levy and Sandler, 2007], is that they are inherently noisy. Firstly, the pages retrieved from a search engine are not guaranteed to be relevant (e.g.,

the word 'Prince' appears in many pages not related to the artist known as Prince). Secondly, the simple fact of appearing together in a Web page does not guarantee that two musical objects are positively associated (e.g., online shops can present their entire music catalogue on the same page). Thirdly, noise further multiplies if search engines are used to find songs rather than artists. In general, the textual nature of this approach makes it difficult to guarantee that the results are indeed representative of any association.

An alternative to Web search analysis is to crawl Internet for entire music collections. Brown et al., 2001 [Brown et al., 2001] designed a system by which users upload their entire music collections to a centralised server, where co-occurrences of songs and artists are calculated. Logan et al., 2003 [Logan et al., 2003] proposed to exploit OpenNap, a popular music sharing service, to obtain personal music collections and infer association between artists under the assumption that artists co-occurring in someone's collection have a better-than-average chance of being similar. These methods are not based on text analysis, thus remove the noise problems related to Web search. The main drawback is that only a limited number of users share their whole music collections on the Internet.

2.2.4 Other ways to assess musical associations

Other approaches not based on experience data can be used to assess whether two songs go well together or not. One option is to analyse intrinsic content and deduce that two songs go well together when they have either similar symbolic representations, similar lyrics or similar acoustic features. These hypotheses, however, are quite strong, since people often relate two songs for cultural and social motivations that cannot be revealed by a content-based analysis. A different option is to ask for experts' opinion.

Symbolic analysis

Chen and Chen, 2001 [Chen and Chen, 2001] first proposed to identify associated songs by comparing their symbolic information in the form of six different features: mean and standard deviation of the pitch values, pitch density, pitch entropy, tempo degree and loudness. Kuo and Shan, 2002 [Kuo and Shan, 2002] proposed to identify associated songs comparing their chords.

The main limitation of these approaches is the requirement of a MIDI [MIDI Manufacturers Association, 1996] representation for each song which is often unavailable and, in the case of polyphonic songs, hardly conveys information about representative tracks. Symbolic information might also be obtained from the score of a theme but, as Logan et al., 2003 [Logan et al., 2003] observed, only a small subset of music has good-quality machine-readable score descriptions available and automatic transcription becomes difficult and error-prone for anything other than monophonic music.

Textual analysis

A different content-based representation for song is by terms of textual context. Kaji et al., 2005 [Kaji et al., 2005] and Mahedero et al., 2005 [Mahedero et al., 2005] first proposed to represent songs as vectors of keywords extracted from the lyrics and to measure their association comparing their weighted cosine coefficients.

Other lyrics-based approaches were proposed in the works by Kleedorfer et al., 2008 [Kleedorfer et al., 2008] and Mayer et al., 2008 [Mayer et al., 2008]: the former uses text mining techniques to identify topic clusters and considers songs falling under the same topic (loss, family, love, etc.) as associated; the latter identifies associated songs based on their rhyme type, part-of-speech and statistic features (e.g., number of 'AABB' rhymes, exclamation marks, articles).

The limitation of these approaches is that they only work for music containing lyrics. Moreover, lyrics have to be available, since automatic lyrics extraction from audio is not yet a reality [Berenzweig and Ellis, 2001].

Acoustic analysis

The most common content-based representation for music is through a series of acoustic features extracted from the audio signal. Aucouturier and Pachet, 2002 [Aucouturier and Pachet, 2002] suggested that associated songs can be identified based on their global timbral quality. A common technique to compute timbral quality is by representing a song in terms of clusters of Mel Frequency Cepstrum Coefficients (MFCCs) [Aucouturier and Pachet, 2004], a measure of spectral shape historically used as a feature for speech recognition: songs are cut into short overlapping frames, a feature vector made of several MFCCs is computed for each frame, a statistical model of the MFCCs' distribution is calculated and finally models are compared to identify songs that sound similar according to their timbre. Approaches based on beat and tempo analysis [Tzanetakis and Cook, 2002] and singing voice segmentation [Berenzweig et al., 2002] have also been investigated to identify songs that sound similar.

Although these unsupervised approaches can interestingly spot out associated songs with different cultural background, or from different genres, they all present a main drawback: scalability. The actual songs (audio content) have to be available to perform the analysis, which limits the results to the music available to each researcher.

Expert analysis

The task to identify associated songs is faced by any disc jockey required to play a sequence of songs that flow smoothly one after the other, for instance in a discotheque, a radio, or a recorded music compilation. Analysing the way in which professional DJs select and order songs in their sets can convey valuable knowledge about associated songs.

Weiss, 2000 [Weiss, 2000] presented a comprehensive analysis on how professional DJs select in real time which songs to play, underlying how the

overall flow and movement between different kinds of music is an important part of the experience. Gates et al., 2006 [Gates et al., 2006] also interviewed several DJs about the way in which musical sequences are prepared.

The expertise of DJs was employed by Ragno et al., 2005 [Ragno et al., 2005] to identify smooth musical transitions. The authors retrieved expertly authored streams from Nielsen Broadcasting Data (a service monitoring the music broadcast in more than 1,600 radio stations, satellite radio and cable music channels), translated them to undirected graphs connecting adjacent songs in each stream, and then used this knowledge to to generate smooth musical playlists.

The main limitation of expert-based analysis is that musical sequences compiled by professionals are scarcely available or are subject to a fee, as for the Nielsen data set.

2.3 Co-occurrence analysis of social data

The purpose of this section is to define a musical association degree that measures *how much* any two songs or artists go well together in sequence.

A social-based approach is employed: first a large set of playlists compiled by thousands of users is retrieved from the Internet, then these playlists are analysed and the most frequently co-occurrent songs and artists identified.

Let $\mathcal{C}$ be the set of songs appearing in the retrieved playlists; the goal of this section is to define a **musical association** degree $s : \mathcal{C}^2 \to [0, 1]$ that measures how much any two songs go well in sequence. The higher the value $s(X, Y)$, the more two songs $X, Y \in \mathcal{C}$ go well together in a musical sequence.

2.3.1 Conditional probability

Let $c(X) \in \mathbb{N}$ and $c(Y) \in \mathbb{N}$ be the **occurrence counts** of song X and Y, that is, the number of playlists where each song occurs; let $d(X, Y, 1) \in \mathbb{N}$ be the number of **contiguous sequential patterns**, that is, the number of times Y occurs in a playlist immediately after X; let

$$N = \sum_{X,Y \in \mathcal{C}} d(X, Y, 1)$$

be the total number of contiguous sequential patterns. The value

$$\frac{d(X, Y, 1)}{c(X)} \tag{2.1}$$

is the conditional probability to find Y as the following element in a playlist containing X. The conditional probability (2.1) looks like a good indicator of the association from X to Y: if people tend to play song Y right after song X, then the value of (2.1) is high, otherwise the value is low. The problem is that this measure is biased by the popularity of Y: it may return a high value

because song Y occurs in many playlists, rather than because X and Y are indeed associated.

For example, let X, Y and Z be three songs with respective occurrence counts $c(X) = 100$, $c(Y) = 60$, and $c(Z) = 5$, and such that $d(X, Y, 1) = 6$ and $d(X, Z, 1) = 4$. The conditional probabilities to find either Y or Z in a playlist after song X are:

$$\frac{d(X, Y, 1)}{c(X)} = 0.06 > \frac{d(X, Z, 1)}{c(X)} = 0.04$$

which show that Y occurs more frequently (6 times) than Z (4 times) as the next song of a sequence after X. Yet, Y is very popular and occurs after X only in *6 out of 60* sequences where it appears (10%), while Z is less popular and still occurs after X in *4 out of 5* sequences where Z appears (80%).

To counter-effect the bias given by the over-popularity of a song (Y in the previous example), the conditional probability (2.1) is multiplied by a weight inversely related to the popularity of the song, as follows:

$$\frac{d(X, Y, 1)}{c(X)} \left(1 - \frac{c(Y)}{N}\right)^{\beta} \tag{2.2}$$

where $\beta \in [0, 1]$ is a parameter tuned to punish more or less the bias given by the individual occurrence count. Note that when $\beta = 0$, the function is identical to (2.1).

The measure (2.2) yields positive values for every pair of songs X and Y that appear in playlists exactly one after the other. There are other pairs of songs that can be considered associated, albeit to a smaller degree. Firstly, two songs X and Y that co-occur in many playlists *separated* by other songs (e.g., X followed by a song, followed by Y) can be seen as associated, since people tend to play them together often, although not exactly one after the other. Secondly, two songs X and Y that co-occur in playlists in the inverse order (X after Y rather than X before Y) can also be seen as associated. Hereafter these cases are integrated in the measure (2.2).

2.3.2 Distant co-occurrences

Let $d(X, Y, D) \in \mathbb{N}$ be the number of playlists where X and Y co-occur at a distance $D \in \mathbb{N}^+$; for example $d(X, Y, 1)$ counts the playlists where Y occurs immediately after X, $d(X, Y, 2)$ counts the playlists where Y occurs after X separated by one song, and so on.

If two songs X and Y occur in playlists not only contiguously but also separated by other songs, this is a further evidence that X and Y are associated. While (2.2) considered only co-occurrences of X and Y at distance 1, the following formula estimates the association considering co-occurrences at a generic distance J:

$$t_J(X, Y) = \frac{d(X, Y, J)}{c(X)} \left(1 - \frac{c(Y)}{N}\right)^{\beta}. \tag{2.3}$$

Co-occurrences between X and Y *at different distances* are aggregated by means of a linear combination that assigns more importance to closer co-occurrences, as follows:

$$\sum_{J=1}^{\delta} \alpha_J t_J(X,Y) \tag{2.4}$$

where $\delta \in \mathbb{N}^+$ is the maximum distance considered and $\alpha_1, \alpha_2, \ldots, \alpha_\delta \in [0,1]$ are the degrees in which distance influences the strength of the association. These values are decreasingly ordered ($\alpha_1 \geqslant \alpha_2 \geqslant \alpha_3 \geqslant \cdots$) to assign more importance to closer co-occurrences, and are such that $\sum_{J=1}^{\delta} \alpha_J = 1$. Note that when $\delta = 1$ the function is identical to (2.2).

2.3.3 Inverse co-occurrences

If two songs X and Y not only occur in many playlists in this order (X before Y), but also in the opposite order (X after Y), this is a further evidence that X and Y are associated. The following measure $t : \mathcal{C}^2 \to [0,1]$ aggregates direct and inverse co-occurrences by means of a linear combination:

$$t(X,Y) = \sum_{J=1}^{\delta} \alpha_J \left[(1-\gamma) \cdot t_J(X,Y) + \gamma \cdot t_J(Y,X)\right] \tag{2.5}$$

where $\gamma \in [0, 0.5]$ is a parameter to take more or less into account the order of occurrence: when $\gamma = 0$ the function is identical to (2.4).

2.3.4 Co-occurrences of songs with artists

So far, only co-occurrences among songs have been examined. Co-occurrences between *artists* are also relevant to determine whether two songs go well in sequence or not. If a song X not only co-occurs often with a song Y but also with other songs by the same artist of Y, this is a further evidence that X and Y are associated.

For example, let X and Y be the songs 'True Colors' (Cyndi Lauper) and 'Holiday' (Madonna). If the data set of playlists is sparse, the number of playlists where X and Y occur together can be small and not disclose the association between the two songs. Observing the co-occurrences between 'True Colors' and other songs by Madonna, though, can reveal that the two artists are indeed related.

Formally, let $a(Y)$ be the artist performing song Y, and let $\mathcal{U}_{X,Y} = \{U \,|\, U \neq Y \wedge a(U) = a(Y) \wedge s(X,U) > 0\}$ be the set of songs from the same artist of Y that co-occur with X. The following measure $u : \mathcal{C}^2 \to [0,1]$ calculates the average degree in which X co-occurs with other songs by the artist of Y:

$$u(X,Y) = \sum_{U \in \mathcal{U}_{X,Y}} \frac{t(X,U)}{\#(\mathcal{U}_{X,Y})} \tag{2.6}$$

where the symbol # denotes the cardinality of a set. The higher the value of $u(X, Y)$, the more X is associated with the artist $a(Y)$, the higher the evidence that X and Y go well together.

2.3.5 Co-occurrences among artists

Even when two songs X and Y do not occur together in playlists, they can be considered associated at a certain level if other songs by their respective artists frequently occur together.

For example, let X and Y be the songs 'Heading For The Moon' (Cyndi Lauper) and 'Supernatural' (Madonna). These two songs might never occur together in a set of playlists, since they are B-sides of rare singles; still other songs by the two pop singers can often co-occur closely, indicating a certain association between X and Y. In general, if many songs by the artist of X occur with many songs by the artist of Y, this is a further evidence that songs X and Y are associated.

Let $\mathcal{V}_{X,Y} = \{V \,|\, V \neq X \wedge a(V) = a(X) \wedge s(V, Y) > 0\}$ be the set of songs from the same artist of X that co-occur with Y. The following measure $v : \mathcal{C}^2 \to [0, 1]$ calculates the average degree in which songs by the artist $a(X)$ other than X co-occur with songs by the artist $a(Y)$ other than Y:

$$v(X, Y) = \sum_{\substack{V \in \mathcal{V} \\ U \in \mathcal{U}}} \frac{t(V, U)}{\#(\mathcal{V}_{X,Y}) \cdot \#(\mathcal{U}_{X,Y})} \,. \tag{2.7}$$

The higher the value of $v(X, Y)$, the more the artist $a(X)$ is associated with the artist $a(Y)$, the higher the evidence that X and Y go well together.

2.3.6 The musical association degree

So far three functions have been defined that estimate some kind of association between songs based on co-occurrences in playlists. The function $t(X, Y)$ estimates the association degree between two songs considering song-to-song co-occurrences (2.5), the function $u(X, Y)$ considers song-to-artist co-occurrences (2.6), and the function $v(X, Y)$ considers artist-to-artist co-occurrences (2.7).

The **musical association degree** $s : \mathcal{C}^2 \to [0, 1]$ between any two songs $X, Y \in \mathcal{C}$ is finally defined combining these three functions by means of a linear combination that assigns more importance to song-to-song co-occurrences, followed by song-to-artist and artist-to-artist co-occurrences, as follows:

$$s(X, Y) = \phi_1 t(X, Y) + \phi_2 u(X, Y) + \phi_3 v(X, Y) \tag{2.8}$$

where the parameters $\phi_1, \phi_2, \phi_3 \in [0, 1]$ are decreasingly ordered ($\phi_1 \geqslant \phi_2 \geqslant \phi_3$) and are such that $\phi_1 + \phi_2 + \phi_3 = 1$.

The function $s(X, Y)$ measures the degree at which two songs sound well together. The fact that $s(X, Y) > s(X, Z)$ implies that song X is better followed in a sequence by song Y than by song Z.

Example 1. Let X, Y and Z be the songs 'True Colors' (Cyndi Lauper), 'Holiday' (Madonna) and 'The Final Countdown' (Europe). To discover whether Y or Z goes better after 'True Colors', a set of playlists is analysed. First, song-to-song associations are considered, comparing the playlists where 'True Colors' and 'Holiday' appear together with the playlists where 'True Colors' and 'The Final Countdown' appear together, and obtaining values for $t(X, Y)$ and $t(X, Z)$. Then, song-to-artist associations are considered, comparing the playlists where 'True Colors' appears with other songs by Madonna and the playlists where 'True Colors' appears with other songs by Europe, and obtaining values for $u(X, Y)$ and $u(X, Z)$. Next, artist-to-artist associations are considered, comparing the playlists where any other song by Cyndi Lauper and Madonna occur together with the playlists where any other song by Cyndi Lauper and Europe occur together, and obtaining values for $v(X, Y)$ and $v(X, Z)$. Finally, the values are aggregated, obtaining the association degrees $s(X, Y)$ and $s(X, Z)$. 'Holiday' results to have a higher degree than 'The Final Countdown' and is therefore identified as the song that goes better after 'True Colors'.

2.4 Working on a real data set

To actually estimate which songs go well in sequence, a repository of playlists is required where to apply the presented co-occurrence analysis. For this purpose, a set of 993,825 playlists has been retrieved from the Web. 85% of the playlists come from the iMix section of the Apple iTunes Store, 10% from the Web community Art of the Mix, and 5% from other Web pages including MusicStrands, Fiql, Go Fish and Webjay. The playlists were retrieved with HTTP calls, incrementally over time, at a rate of about 1,000 per day. Each playlist contains a sequence of IDs to univocally identify songs and artists and does not include information related to the author, title or date of creation of the playlist.

The authors of the playlists are mainly young people from Western countries since this is the typical audience of the harvested communities; for this reason a prevalence of young-targeted, Western music is to be expected. Moreover, some music is not present at all; for instance the playlists retrieved from iTunes Store do not contain songs from The Beatles, since their catalogue is not on sale there.

The assortment of music genres in the playlists is represented in Fig. 2.1. Each point represents a song in the set of playlists, its colour indicates the genre and its position in the plane is calculated with a dimensionality reduction algorithm called Distributed Recursive (Graph) Layout [Martin, 2008] that minimises the distance between the most co-occurrent songs.

The colour distribution highlights the fact that Rock songs are prevalent as they can be found in playlists together with almost any other genre. Other genres, such as Hip Hop/Rap, Jazz and Classical, form compact clusters which indicate that these songs tend to occur in playlists mostly with other songs by the same genres.

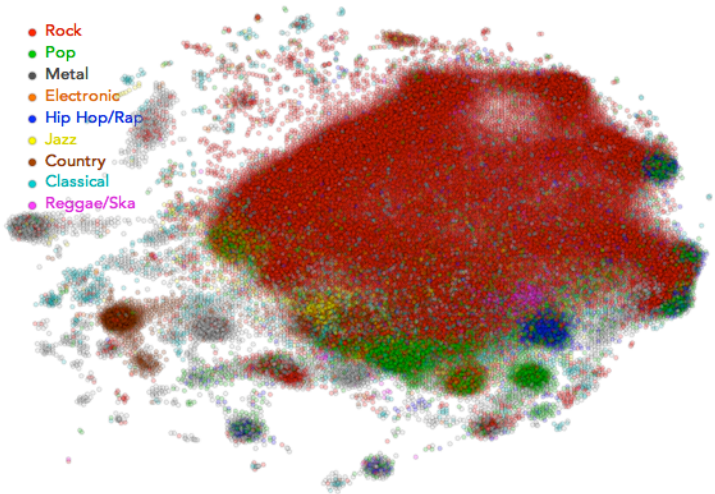

Figure 2.1: Musical associations in the data set of playlists.

2.4.1 Initial data set

The initial data set is made of 993,825 playlists. Playlists where any song or artist is repeated often (more than 4 times) are removed for their co-occurrence analysis would lead to the obvious outcome that a song is associated with itself or with other songs by the same artist.

In the set of remaining playlists, 5,433,413 pairs of songs appear contiguously and are not performed by the same artist. The most common contiguous pairs of songs are: 'Dirty Little Secret' (The All-American Rejects) and 'Dance, Dance' (Fall Out Boy); 'Grillz' (Nelly) and 'Laffy Taffy' (D4L); 'Boulevard Of Broken Dreams' (Green Day) and 'Mr Brightside' (The Killers); 'My Humps' (Black Eyed Peas) and 'Run It!' (Chris Brown); 'Caring Is Creepy' (The Shins) and 'In The Waiting Line' (Zero 7); 'Don't Cha' (Pussycat Dolls) and 'Pon De Replay' (Rihanna).

One problem with this data set is that it contains both 'good' and 'bad' playlists. People are free to publish on the Web any type of playlists and there is no implicit guarantee that the collected ones are made of songs with an affinity, as listeners can as well create and share random sets of tracks without any meaningful order. To obtain valid musical associations, bad playlists should be removed from the data set before proceeding with the co-occurrence analysis.

2.4.2 Noise filtering process

Two hypotheses are made for filtering out bad playlists. The first is that any playlist containing alphabetically ordered songs or artists were probably not compiled with a specific listening purpose (e.g., relaxing, jogging, partying) but as mere groups of consecutive tracks. The second hypothesis is that neither very short nor very long playlists were created with a purpose that is coherent with the proposed interpretation of playlists. If these playlists were discarded, then the data set would be reduced to *less* playlists with a *high quality*, that is, playlists which actually contain consecutive associated songs.

The noise filtering process consists in removing from the initial data set any playlist that has less than five songs, more than twenty songs, or has more than five alphabetically ordered songs or artists. These specific values were determined after having observed the distribution of alphabetically ordered songs and artists and the distribution of playlist lengths in the initial data set, as illustrated in Fig. 2.2.

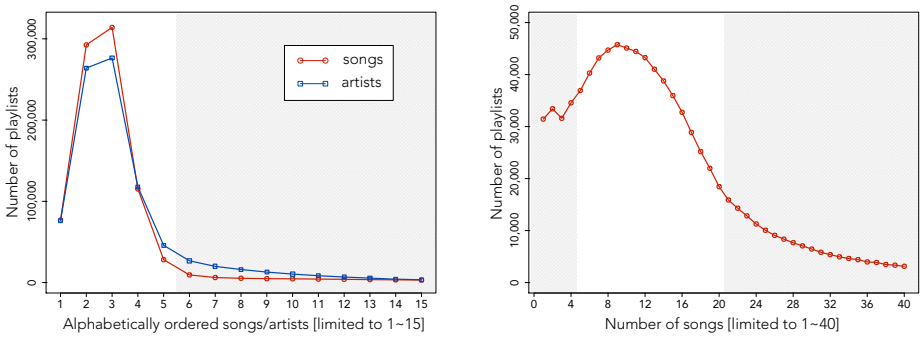

Figure 2.2: Number of playlists with alphabetically ordered songs/artists (left) and with specific lengths (right).

The noise filtering process reduces the data set from 993,825 to 465,438 playlists. The number of contiguous pairs of songs also decreases from 5,433,413 to 1,256,681.

2.5 Tuning parameters

The next step before applying the co-occurrence analysis process to the filtered set of playlists is to decide the values of different parameters introduced in Sect. 2.3. The parameters are δ, α_J and γ, which occur in the functions (2.5) and (2.8). These parameters determine how much the distance and the order of songs in playlists influence their association.

2.5.1 Tuning the distance parameters

The maximum distance parameter is set to $\delta = 3$, which stands as a compromise between contiguous and distant co-occurrences. This means that two songs that appear in the same playlist separated by less than three songs are considered as associated, otherwise the association does not subsist.

The distance parameters $\alpha_1, \alpha_2, \alpha_3 \in [0, 1]$ determine the degree in which closer co-occurrences are more relevant than distant ones. These parameters are set to $\alpha_1 = 0.6, \alpha_2 = 0.3, \alpha_3 = 0.1$ to gradually assign higher importance to closer co-occurrences.

2.5.2 Tuning the order parameter

The parameter $\gamma \in [0, 0.5]$ determines the degree in which inverse co-occurrences (X *after* Y) also contribute to determine their associations. In this case, this parameter is set to $\gamma = 0.2$: direct co-occurrences (where X occurs *before* Y) are identified as more relevant to determine the association between X and Y than inverse co-occurrences.

Setting the parameter to $\gamma = 0.2$ expresses the fact that the order in which two songs occur in playlists has an effect on the measure of their association. This value was decided observing that many playlists in the data set are *asymmetric* by nature: songs sound well together in one direction but not necessarily in the other one. This can be explained by the fact that several authors, especially disc jockeys, compile playlists where the *end* of each song mixes with the *beginning* of the next one. In these cases, the order should be accounted for when measuring associations.

2.6 The resulting associations

Having filtered noise from the initial data set of 993,825 playlists and having applied the co-occurrence analysis technique with the parameters set to:

- maximum distance between songs: $\delta = 3$,
- degrees in which distance influences the strength of the association: $\alpha_1 = 0.6, \alpha_2 = 0.3, \alpha_3 = 0.1$,
- degree in which popularity bias is punished: $\beta = 0.8$,
- degree in which order influences the association: $\gamma = 0.2$,
- degrees in which different types of co-occurrences influence the strength of the association: $\phi_1 = 0.6, \phi_2 = 0.3, \phi_3 = 0.1$,

the association degrees $s(X, Y)$ defined in Sect. 2.3 are finally calculated.

Associations between songs by the same artist are excluded for their obviousness, as well as songs by 'virtual' artists (such as 'Various Artists' or 'Soundtrack') and songs appearing in less than 5 playlists, for their minimal

statistical significance. This threshold was fixed observing how the popularity of songs distributes in the data set of playlists (Fig. 2.3).

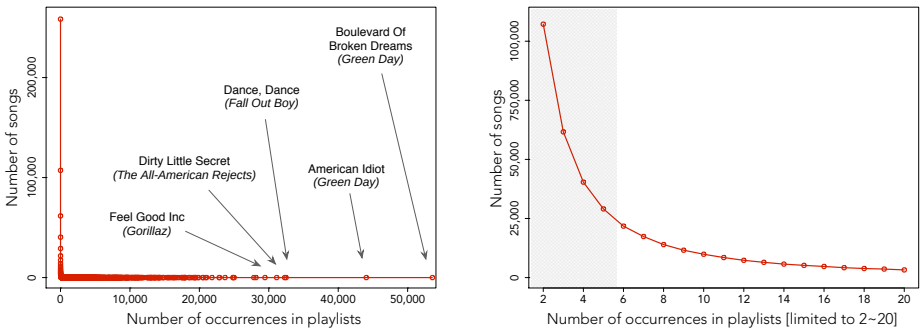

Figure 2.3: Occurrences per songs in retrieved playlists.

Eventually, a musical association degree $s(X, Y)$ was estimated for as many as 46,217,300 distinct pairs of 415,498 songs by 47,148 distinct artists.

2.6.1 Evaluating song associations

Given a song X in the data set, the songs Y with the highest association degree $s(X, Y)$ are the songs that go best in a sequence after X according to the human experience recorded in playlists. The following is an example of list of top associated tracks.

Example 2. The song 'New York, New York' (Frank Sinatra) occurs in 219 playlists. The analysis finds that 173 distinct songs appear in playlists with 'New York, New York', that 6,458 songs appear in playlists with other songs by Frank Sinatra, and that 26,402 songs are from artists appearing in playlists with songs by Frank Sinatra. The co-occurrence analysis allows to rank all these songs according to how much they are associated with 'New York, New York', uncovering the following songs as the top associated ones: 'The Waters Of March' (Susannah McCorkle), 'Stardust' (Glenn Miller), 'New York's My Home' (Sammy Davis Jr.), 'Manhattan Avenue' (Nellie McKay), 'Mississippi Goddam' (Nina Simone), 'Something Beautiful' (Robbie Williams).

The example highlights some common properties of the lists of top associated songs. The first characteristic is that the top songs are not particularly famous, yet share a strong affinity with the seed track. 'The Waters Of March' (Susannah McCorkle), for instance, is strongly connected with 'New York, New York' (Frank Sinatra) since they were both composed in the Seventies and performed as musical standards by various singers. The reason why uncommon songs appear higher in the list is the high value of the popularity bias parameter $\beta = 0.8$ which punishes songs that occur very often in the set of playlists.

Another positive characteristic of the list is diversity: all the songs are performed by different artists and belong to multiple genres and periods;

'Something Beautiful' (Robbie Williams), for instance, was published in 2003.

2.6.2 Evaluating artist association

As well as lists of top associated tracks, the co-occurrence analysis can produce lists of top associated artists. Given an artist $a(X)$ in the data set, the artists whose songs have in average the highest association degree with the songs of $a(X)$ are the top associated artists with $a(X)$.

Working at the level of artists is not as specific as working at the level of songs: two songs can go well together for acoustic reasons (same timbre, rhythm, lyrics) while a correlation between two artists is more abstract, implying some sort of social or cultural relationships (e.g., two artists are contemporary, play the same instrument, have collaborated, belong to the same genre). An example of list of top associated artists is reported hereafter.

Example 3. Themes written by the soundtrack composer John Williams appear in 2,367 playlists. The analysis finds out that songs by 866 distinct artists appear in playlists with any track by John Williams. These artists are ranked according to how much they are musically associated with John Williams; the following artists are uncovered as the most associated: Itzhak Perlman, Christopher Young, Arthur Fiedler, London Symphony Orchestra, John Debney, Danny Elfman, John Carpenter, John Barry.

Similarly to Example 2, an interesting property of this list is that uncommon but strongly associated items appear first, followed by more popular ones. Itzhak Perlman, for instance, is not a soundtrack composer but a violinist who played first violin in 'Schindler's List' soundtrack, composed by John Williams. Arthur Fiedler also shares a strong affinity with John Williams, who took his place as the conductor of the Boston Pops orchestra in 1979 after Fiedler's death. John Debney, Danny Elfman and John Barry are award-winning movie composers contemporary to John Williams.

2.6.3 Comparisons with other music similarity measures

For the purpose of evaluation, some lists of top associated songs and artists are hereafter contrasted with equivalent lists calculated with distinct sources of musical knowledge obtained from different Web sites. The Web sources used for the comparison are: Yahoo! Music, Last.fm and Audiobaba for associated songs and All Music Guide, Yahoo! Music, Last.fm, MusicStrands and MusicSeer for associated artists. The technique presented in this chapter does not exactly look for 'similar' songs, but for songs that go well together in sequence. Nevertheless, comparing its assessments with those offered by other techniques can reveal interesting peculiarities.

Yahoo! Music makes available for each song and artist in its catalogue a list of similar tracks and artists "generated from end user feedback" [Baumann and Halloran, 2004]. Last.fm offers for each song a Web page listing what people who

listened to that song also listened to. Audiobaba uses a content-based approach: for each song, a list of tracks that are acoustically similar is returned.

For associations among artists, All Music Guide includes handcrafted contributions written by expert editors and is considered as the 'bible' of music reviews, the ground truth for music classification research [Pachet and Cazaly, 2000; Hu and Downie, 2007; Magno and Sable, 2008; Sordo et al., 2008]. MusicStrands derives associations from the same data set of playlists employed in this dissertation, but applying a different technique based on "bags of associations" [Shur and Hangartner, 2006]. MusicSeer follows two different approaches: one is based on a survey to collect subjective judgements about artist similarity, the other one on playlist co-occurrences. The MusicSeer survey includes associations for about 413 artists [Logan et al., 2003], while the playlist-based approach reports associations between 60,931 artists from the analysis of 29,164 playlists retrieved from Art Of The Mix [Ellis, 2003].

Tables 2.1, 2.2 and 2.3 report the most associated tracks for the songs 'New York, New York' (Frank Sinatra), 'Whenever, Wherever' (Shakira) and 'Smoke On The Water' (Deep Purple). Tables 2.4, 2.5, 2.6 and 2.7 compare the top artists delivered for John Williams, Frank Sinatra, Abba and Destiny's Child.

Table 2.1: Top associated songs for 'New York, New York' (Frank Sinatra).

Poolcasting	'The Waters Of March' (Susannah McCorkle), 'Stardust' (Glenn Miller), 'New York's My Home' (Sammy Davis Jr.), 'Manhattan Avenue' (Nellie McKay), 'Mississippi Goddam' (Nina Simone), 'Something Beautiful' (Robbie Williams)
Yahoo!	'Piano Man' (Billy Joel), 'That's Amore' (Dean Martin), 'At Last' (Etta James), 'Mrs. Robinson' (Simon & Garfunkel), 'The Boxer' (Simon & Garfunkel), 'Bridge Over Troubled Water' (Simon & Garfunkel)
Last.fm	'Strangers In The Night' (Frank Sinatra), 'Fly Me To The Moon' (Frank Sinatra), 'What A Wonderful World' (Louis Armstrong), 'Walkin' My Baby Back Home' (Nat King Cole), 'Try To Remember' (Bobby Darin), 'Beyond The Sea' (Bobby Darin)
Audiobaba	'Woman, Woman' (Gary Puckett), 'Spirit In The Night' (Bruce Springsteen), 'You Make Me Feel So Young' (Frank Sinatra), 'Mud On The Tires' (Brad Paisley), 'Lucky To Be Here' (Montgomery Gentry), '6 AM' (Random Access)

The analysis of these lists of top associated songs does not reveal a great diversity among different musical sources, although some peculiarities can be observed. As noted previously, co-occurrence analysis of playlists can identify songs that are associated but not overly popular, something that does not happen with every other technique. Yahoo! Music, for instance, returns the famous hit 'Baby One More Time' (Britney Spears) as the top associated song for 'Whenever, Wherever' (Shakira). For this song, 'You Spin Me Round' (Thalía) is possibly a better match since both songs were released in 2001 and performed by Latin American singers (see Table 2.2). Another observation is that Last.fm

Table 2.2: Top associated songs for 'Whenever, Wherever' (Shakira).

Poolcasting	'You Spin Me Round' (Thalía), 'Crazy In Love' (Beyoncé & Jay-Z), 'Grazing In the Grass' (Raven-Symoné), 'Si Ya Se Acabó' (Jennifer Lopez), 'Freakout' (B*Witched), 'Perros' (Paulina Rubio)
Yahoo!	'Baby One More Time' (Britney Spears), 'Irresistible' (Jessica Simpson), 'If You Had My Love' (Jennifer Lopez), 'Candy' (Mandy Moore), 'Bye Bye Bye' ('N Sync), 'Genie In A Bottle' (Christina Aguilera)
Last.fm	'Underneath Your Clothes' (Shakira), 'Objection (Tango)' (Shakira), 'Radar' (Britney Spears), 'Keeps Gettin' Better' (Christina Aguilera), 'Beautiful Liar' (Beyoncé feat. Shakira), 'Waiting for Tonight' (Jennifer Lopez)
Audiobaba	'Too Bad' (Bad Company), 'Sunday Girl' (Erasure), 'Life In The Fast Lane' (Eagles), 'These Dreams Of You Are So Much Sweeter Than The Truth' (The Sharp Things), 'Powerful Thing' (Trisha Yearwood), 'Migration' (Jimmy Buffett)

Table 2.3: Top associated songs for 'Smoke On The Water' (Deep Purple).

Poolcasting	'Silver Machine' (Hawkwind), 'The Joker' (Steve Miller Tribute Band), 'Dream Evil' (Dio), 'Rock N' Roll Hoochie Koo' (Johnny Winter), 'South Saturn Delta' (Jimi Hendrix), 'Nottingham Lace' (Buckethead)
Yahoo!	'Melissa' (The Allman Brothers Band), 'Surrender' (Cheap Trick), 'Sweet Talkin' Woman' (Electric Light Orchestra), 'Somebody To Love' (Jefferson Airplane), 'White Rabbit' (Jefferson Airplane), 'Maggie May' (Maggie May)
Last.fm	'Highway Star' (Deep Purple), 'Child in Time' (Deep Purple), 'Stairway to Heaven' (Led Zeppelin), 'Paranoid' (Black Sabbath), 'Whole Lotta Love' (Led Zeppelin), 'Black Dog' (Led Zeppelin)
Audiobaba	'Double E' (Neil Young), 'This Can't Be Love' (Freeborn), 'The Lady' (TheHookUp), 'Elegant People' (Jaco Pastorius Big Band), 'Last Chance' (Ear Candy), 'Diggers Of Ditches Everywhere' (These Arms Are Snakes)

Table 2.4: Top associated artists for John Williams.

Poolcasting	Itzhak Perlman, Christopher Young, Arthur Fiedler, London Symphony Orchestra, John Debney, Danny Elfman, John Carpenter, John Barry
MusicStrands	Danny Elfman, Vangelis, Hollywood Studio Orchestra, Erich Kunzel, Green Day, Gorillaz, Weird Al Yankovic, John Barry, Queen, Eminem
AllMusic	John Barry, Jerry Goldsmith, Elmer Bernstein, Howard Shore, Erich Korngold
Yahoo!	Franz Joseph Haydn, James Newton Howard, Michael Kamen, National Philarmonic Orchestra, Alan Silvestri, Jerry Goldsmith, John Barry
Last.fm	Jerry Goldsmith, James Horner, Patrick Doyle, Alan Silvestri, James Newton Howard, Howard Shore, Hans Zimmer, Nicholas Hooper, Danny Elfman

Table 2.5: Top associated artists for Frank Sinatra.

Poolcasting	Dean Martin, Sammy Davis Jr., Judy Garland, Bing Crosby, The California Raisins, Tony Bennett, Louis Prima, Rosemary Clooney, Nat King Cole
MusicStrands	Dean Martin, Billie Holiday, Nat King Cole, Perry Como, Ella Fitzgerald, Andy Williams, Tony Bennett, Etta James, Bing Crosby, Diana Krall
AllMusic	Dean Martin, Vic Damone, Dick Haymes, Sarah Vaughan, Nat King Cole, Dinah Washington, Mel Tormé, Ella Fitzgerald, Tony Bennett, Jo Stafford
Yahoo!	Dean Martin, Tony Bennett, Nat King Cole, Ray Charles, The Beach Boys, Simon & Garfunkel, Elvis Presley, The Beatles, Norah Jones, Norah Jones
Last.fm	Dean Martin, Sammy Davis, Jr., Frank Sinatra & Tommy Dorsey, Tony Bennett, Nat King Cole, Bobby Darin, Ella Fitzgerald, Mel Tormé
MS Survey	Eric Clapton, Billy Joel, Elton John, Elvis Costello, Elvis Presley, Van Morrison, John Lennon, Bob Dylan, Nine Days, Ozzy Osbourne
MS Playlists	Elvis Presley, Elton John, John Denver, Abba, Whiskeytown, Beatles, Billy Joel, Bob Marley, Eric Clapton, Everly Brothers

Table 2.6: Top associated artists for Abba.

Poolcasting	Agnetha Fältskog, A-Teens, Chic, Gloria Gaynor, The 5th Dimension, Andy Gibb, Olivia Newton-John, Rose Royce, KC & The Sunshine Band, Bee Gees
MusicStrands	Donna Summer, Madonna, Gloria Gaynor, Cyndi Lauper, Blondie, Kool & The Gang, Elton John, The B-52s, Michael Jackson, Diana Ross
AllMusic	Ambsoris, Olivia Newton-John, The Carpenters, Captain & Tennille, Bucks Fizz, Fletwood Mac, Andy Gibb, Lindsey Buckingham, The Cowsills
Yahoo!	Bee Gees, The Carpenters, Elvis Presley, The Beatles, Foreigner, Whitney Houston, Bon Jovi, Madonna, Barry Manilow, Michael Jackson
Last.fm	Agnetha Fältskog, Frida, Boney M., Bee Gees, Olivia Newton-John, Baccara, Cher, Bucks Fizz, Donna Summer, Army of Lovers, Alcazar
MS Survey	Ace of Base, Bee Gees, Blondie, Spice Girls, Olivia Newton-John, Beach Boys, Roxette, Cyndi Lauper, Backstreet Boys, Donna Summer
MS Playlists	Bee Gees, Blondie, Cyndi Lauper, Queen, Cat Stevens, Cher, Beach Boys, Donna Summer, Olivia Newton-John, Phil Collins

Table 2.7: Top associated artists for Destiny's Child.

Poolcasting	Kelly Rowland, City High, Ciara, Fantasia, Christina Milian, Beyoncé, Ashanti, Girls Aloud, 3LW, Dru Hill
MusicStrands	Ciara, Pussycat Dolls, Usher, Beyoncé, Nelly, 50 Cent, Mariah Carey, Chris Brown, Gwen Stefani, Eminem
AllMusic	Toni Braxton, Mariah Carey, Jennifer Lopez, Aaliyah, Xscape, Ginuwine, Deborah Cox, Kelly Price, Faith Evans, Brandy, Usher, Mya
Yahoo!	Faith Evans, Cruel Story Of Youth, Nich Lachey, Jamie Foxx, Jessica Simpson, Ciara, Jagged Edge, Lil' Kim, Ryan Cabrera, Janet Jackson
Last.fm	Beyoncé, Kelly Rowland, Michelle Williams, LeToya, Solange, Ashanti, Ciara, Brandy, Mariah Carey, Monica, Aaliyah, TLC, Mya

and Yahoo! Music have limited diversity in the results: Last.fm returns two tracks by Frank Sinatra as the top associated for 'New York, New York' (see Table 2.1) while Yahoo! Music repeats songs by the same artists (Simon & Garfunkel in Table 2.1 and Jefferson Airplane in Table 2.3).

The analysis of top associated artists also reveals particular behaviours for the different methods. First, almost every source of knowledge yields Dean Martin as the top associated artist with Frank Sinatra (see Table 2.5), marking a clear affinity between the two. Next, the playlist-based approach introduced in this chapter returns first artists that are not very popular but strongly associated.

An example is provided by Table 2.6 where Agnetha Fältskog shows up as the top associated artist for Abba according to the co-occurrence analysis of playlists. Agnetha Fältskog is not a very famous solo artist but she is indeed popularly known as the lead singer of Abba. Similarly, Table 2.7 shows Kelly Rowland (one of the members of Destiny's Child) as one of the top associated artist with Destiny's Child.

This behaviour is possibly the most distinct advantage of the playlist-based method: to be able to spot out items that share a strong social affinity. This result is obtained from the automatic analysis of playlists, without human intervention. All Music Guide, on the other hand, requires man-hours of dedicated expert work to obtain results that are almost equivalent in terms of affinity and sometimes worse in terms of diversity.

2.7 Summary

This chapter has presented a technique to measure the associations that exist between any two songs and artists. Associations are estimated observing how millions of persons organise music for their daily activities.

The technique consists in collecting a large amount of playlists from the Web and analysing the co-occurrences of songs, under the idea that the more two songs occur together and closely, the stronger their association.

Playlists are expressions of human experiences and include factual knowledge about how people listen to music. Playlists are ordered sequences of songs associated for acoustic, cultural or social reasons that cannot be completely uncovered by experts or content-based methods.

The proposed method analyses, for each pair of songs (X, Y), the number of playlists where they both appear, their order, distance, popularity and the co-occurrences of their artists and obtains a measure $s(X, Y) \in [0, 1]$ of their musical association.

This method has been applied to a real data set of about a million playlists retrieved from multiple Web-based music communities. The result has been the estimation of musical associations for 415,498 distinct songs from 47,148 artists. For each song and artist, a list of top associated items can be compiled. These lists tend to show first items that are not very popular but strongly associated with the seed item, followed by more popular ones.

Some of these lists have been compared with equivalent ones gathered from music-related Web services such as All Music Guide and Last.fm. The comparison has shown results that are quite equivalent independently of the source of knowledge used. The main advantages of the automatic technique introduced in this chapter are its scalability and the ability to spot out artists with a strong social affinity (Agnetha Fältskog for Abba, Kelly Rowland for Destiny's Child), a property not found in the expert-based knowledge of All Music Guide.

The musical associations estimated in this chapter will be employed in the poolcasting technique introduced in Chap. 4 to determine which songs to play in a musical sequence to guarantee a sense of continuity from one song to the next.

Chapter 3

Individual listening behaviours

I know just what you need
I might just have the thing

Soul Asylum, 1995

3.1 Music libraries and listening habits

Different persons are characterised by different listening behaviours: some are used to play all kinds of music in a shuffled order, others to listen only to a particular genre; some are eager to discover the latest trends of contemporary music, others are obsessed with playing the same album again and again.

The previous chapter focused on music organised in playlists. Some persons do not use playlists at all but decide in real time which songs to play in their music devices. The simple fact of a person picking a particular song to play is already a musical experience that is worth be analysed.

The wide proliferation of digital music players has made it easy to track the history of a listener. Most digital players (Apple iTunes, iPod, Winamp, Windows Media Player, etc.) store data about each played song; Apple iTunes, for instance, records the play count (number of times the song was played), the play recency (last time the song was played), the skip count (number of times the song was skipped before its end) and the user rating for each song in the music library (see Fig. 3.1).

Listening behaviour data is very descriptive of personal preferences: having played certain songs and not other ones outlines the musical taste of an individual better than a verbal description, with its inaccuracies and misinterpretations, would.

As many persons share their listening behaviour data on the Internet to make their friends aware of their musical habits, the Web has become the best place

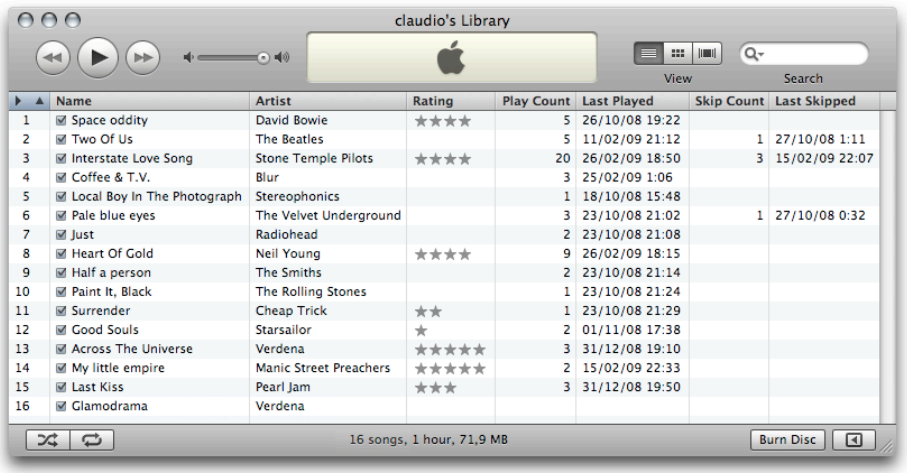

Figure 3.1: Personal music library managed with iTunes.

where to find data about which songs a person has played, when, how they were rated, et cetera.

The purpose of this chapter is to introduce a measure of **musical preference** that automatically estimates how much a person likes a given song based on personal listening habits data collected from the Web.

3.2 Previous work

The issue faced in this chapter consists in identifying a set of individual musical preferences without having to explicitly ask for them. The process of acquiring user preferences is known as user modelling [Rich, 1979; Kass and Finin, 1988; Kobsa, 2001] and can either be explicit or implicit.

3.2.1 Explicit user modelling

Explicit acquisition of preferences is obtained when individuals *actively* provide specific facts about loved items. Most online stores, for instance, ask their users to provide feedback about purchased items: Amazon collects ratings about books, Trip Advisor about travel destinations, eBay about sellers' reputation.

Explicit feedback is the most direct way for users to express their taste, although Claypool et al., 2001 [Claypool et al., 2001] noticed how having to stop to enter explicit ratings can alter normal patterns of information usage, and Zhang et al., 2002 [Zhang et al., 2002] proved how users can rapidly stop providing explicit ratings unless they perceive a benefit. Amazon and eBay, for instance, gratify frequent raters highlighting their profiles as 'experts' within the community of users.

Potter, 2008 [Potter, 2008] also pointed out how explicit statements require users to 'translate' their inner preferences into a score, which is not an immediate and unequivocal process: deciding how many 'stars' to assign to a good item is a matter of personal interpretation. Banerjee, 1992 [Banerjee, 1992] also noticed that users asked to rate an item are easily influenced by the scorings assigned by previous users.

3.2.2 Implicit user modelling

Implicit user modelling takes place by *observing* user actions and inferring preferences from the observed behaviours. For instance, someone always buying clothes in the same store implicitly expresses a preference for that fashion style. Similarly, forwarding an online video to ten friends implicitly demonstrates an interest for that video.

In different publications, Nichols, 1997 [Nichols, 1997], Oard and Kim, 2001 [Oard and Kim, 2001] and Kelly and Teevan, 2003 [Kelly and Teevan, 2003] have categorised several types of actions as possible sources for implicit user modelling. Figure 3.2 reports the list of actions according to the underlying purpose of the observed action (Behaviour Category), and to the smallest possible scope of the item being acted upon (Minimum Scope). In the domain of music, for instance, having either purchased, saved, rated or listened a specific song can be interpreted as an implicit preference for that song.

		Minimum Scope		
		Segment	Object	Class
Behaviour Category	Examine	View, Listen, Scroll, Find, Query	Select	Browse
	Retain	Print	Bookmark, Purchase, Delete, Save, Email	Subscribe
	Reference	Copy-and-paste, Quote	Forward, Reply, Link, Cite	
	Annotate	Mark up	Rate, Publish	Organise
	Create	Type, Edit	Author	

Figure 3.2: Classification of behaviours used to infer implicit feedback.

3.2.3 Building user profiles

Collecting implicit or explicit preferences is only the first step to compile a comprehensive model of user preferences. As pointed out by Hofmann, 2004 [Hofmann, 2004], different users associate different meanings with ratings; for instance, '4 out of 5 stars' may have a different meaning for different people. For this reason, individual preferences need be *normalised* in order to be compared.

Goldberg et al., 2001 [Goldberg et al., 2001] proposed to normalise each rating by subtracting its mean rating over all users, and then dividing by its standard deviation.

Another relevant aspect of user modelling is how to build a comprehensive model when only a few individual preferences are available. Dieterich et al., 1993 [Dieterich et al., 1993] suggested to either start with an *empty* model, to classify the user into one *stereotypical* category, or to build an initial *individual* model based on a preliminary question-answering session. A stereotypical approach, for instance, is used by Last.fm which uses geographical position obtained from the IP address of unregistered users to recommend musical events taking place in the area where users are located.

Interests may change along time, so a system that learns a model of the user's interests should be based on algorithms that can quickly adjust to changing interests. The notion of changing target concepts is known as "concept drift" [Billsus and Pazzani, 2007], or "persistence of interest" [Lieberman, 1995]. Dieterich et al., 1993 [Dieterich et al., 1993] distinguished between "short-term" data — user preferences that are valid only for the current context or session — and "long-term" data — which should be kept beyond the current session and saved in a permanent storage medium. In the domain of music, for instance, Last.fm online radio differentiates between 'skipping' a song (short-term negative preference) and 'banning' a song (long-term negative preference) from a personalised music channel.

3.3 Gathering listening habits

To determine individual music preferences from the analysis of personal listening habits, these have to be collected first, which can occur in two ways.

The first method is to extract this information directly from personal music libraries. Apple iTunes, for instance, stores in the hard disk a file called 'iTunes Library.xml' which describes the list of played songs, play counts, play recency, skip counts, user ratings. Parsing this XML file provides first-hand knowledge about the music played on that particular machine.

The second method is to gather this knowledge from the Internet. Different tools track personal listening habits and make these data available on the Web. Last.fm, for instance, provides each member with a Web page showing the last songs played, the device where they were played and the assigned rating, as detected by the music tracking software Audioscrobbler (see Fig. 3.3). MusicStrands offers a similar service with a tool called MyStrands (see Fig. 3.4).

Gathering listening habits from the Web has four main advantages. Firstly, data can be collected from thousands of users without having to access their hard disks. Secondly, Web services provide Web Application Programming Interfaces (API) [Booth et al., 2004] to rapidly retrieve large amount of data in a standard XML format. Thirdly, Web services can automatically fix mistakes in song titles (e.g., 'Obladi Oblada' rather than 'Ob-la-di Ob-la-da') thus correctly

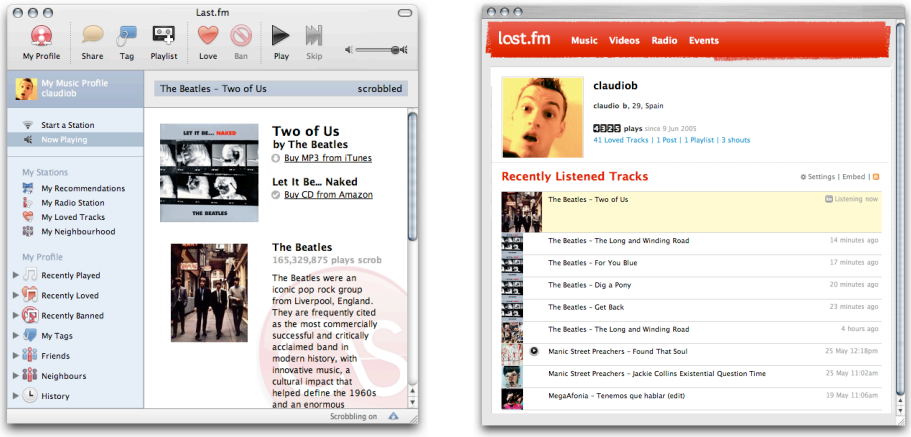

Figure 3.3: Last.fm tracking tool Audioscrobbler and profile Web page.

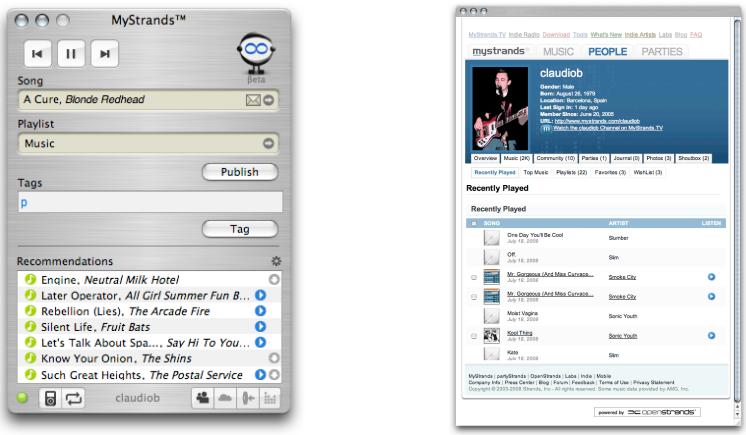

Figure 3.4: MusicStrands tracking tool MyStrands and profile Web page.

identifying played songs. Fourthly, Web services can automatically aggregate the listening habits from multiple devices; for instance Last.fm 'scrobbles' both songs played in portable devices (iPod, iPhone, Android) and digital libraries (iTunes, Windows Media Player, Winamp).

3.4 Estimating individual preferences

Once listening habits data have been collected, a choice has to be made about which part of these data can be considered as illustrative of music preferences.

One relevant property is clearly the **user rating** assigned to each song. Almost every digital player enables listener to vote for favourite songs; Apple iTunes uses 'stars' (from 1 to 5), while Audioscrobbler gives the option to mark a song as either 'Loved' or 'Banned'.

A second relevant property is the **play count**: how many times a song was played. Intuitively, the interest of a person for a song is directly related to its play count, especially if the song has been played often recently.

Other relevant properties, such as the skip count and the play recency, will not be utilised since they are more subject to interpretation. Skipping a song, for instance, might be interpreted as a negative preference, but listeners also skip songs they like when these do not fit in the current context.

The rest of this section explains how, given the ratings assigned by a person to a set of songs and their play counts, a measure of **individual preference** for each song can be defined.

3.4.1 Usage behaviour normalisation

Let $\mathcal{U}$ be a group of people and let $U \in \mathcal{U}$ be a person for which listening habits data are available with respect to a set of songs $\mathcal{C}$. Let $r : \mathcal{U} \times \mathcal{C} \to [\varrho_{\min}, \varrho_{\max}]$ be the rating assigned by U to each song, where $\varrho_{\min}$ and $\varrho_{\max}$ are the minimum and maximum rating scores (e.g., $\varrho_{\min} = 1$ and $\varrho_{\max} = 5$ 'stars' in Apple iTunes), and let $n : \mathcal{U} \times \mathcal{C} \to \mathbb{N}$ be the play count, that is, the number of times that U played each song.

The goal is to combine for each song $X \in \mathcal{C}$ the two values $r(U, X)$ and $n(U, X)$ in order to measure the **individual preference** degree $i : \mathcal{U} \times \mathcal{C} \to [0, 1]$, that is, how much U shows to like song X.

High ratings and high play counts identify songs a listener likes. The 'absolute' values of rating and play count, though, offer small information to estimate a preference degree. Having listened to a song 3 times or having assigned a rating of '3 out of 5' stars, for instance, do not clearly indicate whether a listener likes a song or not. If that listener normally rates songs with only 1 or 2 stars, then giving 3 stars expresses a positive preference. If the average rating is instead 4 or 5 stars, then 3 stars is probably not a good signal. Similarly, a play count of 3 is significant or not whether the listener is a sporadic or a frequent music listener.

User ratings and play counts are values that are relevant only with respect to the *average listening behaviour*. Only songs showing a play count or a rating above the average can significantly be considered as favourite songs. For this reason, user ratings $r(U, X)$ and play counts $n(U, X)$ are hereafter *normalised* to yield values in the range $[-1, 1]$, so that only songs 'above the average' are assigned positive values.

3.4.2 Normalising user rating

To define a *normalised user rating* means to find a function $\widehat{r} : \mathcal{U} \times \mathcal{C} \to [-1, 1]$ which returns positive (resp., negative, null) values for songs rated higher than (resp., lower than, equal to) the average. Let $\mathcal{R}_U \subseteq \mathcal{C}$ be the songs that a person $U \in \mathcal{U}$ has ever rated and let

$$\overline{r(U)} = \frac{\sum_{X \in \mathcal{R}_U} r(U, X)}{\#(\mathcal{R}_U)}$$

be the average user rating of U. The normalised user rating $\widehat{r}(U, X)$ is a function that satisfies these conditions:

$$\begin{cases} r(U,X) < \overline{r(U)} & \Longrightarrow \quad -1 < \widehat{r}(U,X) < 0 \\ r(U,X) = \overline{r(U)} & \Longrightarrow \quad \widehat{r}(U,X) = 0 \\ \overline{r(U)} < r(U,X) & \Longrightarrow \quad 0 < \widehat{r}(U,X) < 1 \end{cases} \tag{3.1}$$

and such that any song with the lowest (resp., highest) possible rating obtains the minimum (resp., maximum) possible normalised value:

$$\begin{cases} r(U,X) = \varrho_{\min} & \Longrightarrow \quad \widehat{r}(U,X) = -1 \\ r(U,X) = \varrho_{\max} & \Longrightarrow \quad \widehat{r}(U,X) = 1 \end{cases} \tag{3.2}$$

under the condition that $\varrho_{\min} < \overline{r(U)} < \varrho_{\max}$.

One function that fulfils (3.1) and (3.2) and is furthermore continuous, monotonic, derivable, with the first derivative monotonic in the interval $(-1, 1)$ is the function that solves the following linear equation system:

$$\begin{cases} a \log(b + \varrho_{\min}) + c = -1 \\ a \log(b + \overline{r(U)}) + c = 0 \\ a \log(b + \varrho_{\max}) + c = 1 \end{cases}$$

and that yields 0 when $r(U,X) = \overline{r(U)}$. The solution of this system is defined by cases:

$$\widehat{r}(U,X) = \begin{cases} 0 & \text{if } r(U,X) = \overline{r(U)} \\ \dfrac{2(r(U,X) - {\varrho_{\text{med}}}^2)}{\varrho_{\max} - \varrho_{\min}} & \text{if } \overline{r(U)} = \varrho_{\text{med}} \\ \dfrac{\log \dfrac{2r(U,X)(\varrho_{\text{med}} - \overline{r(U)}) + \overline{r(U)}^2 - \varrho_{\max}\varrho_{\min}}{(\varrho_{\max} - \overline{r(U)})(\overline{r(U)} - \varrho_{\min})}}{\log(\varrho_{\max} - \overline{r(U)}) - \log(\overline{r(U)} - \varrho_{\min})} & \text{otherwise,} \end{cases}$$

where $\varrho_{\text{med}} = \frac{1}{2}(\varrho_{\min} + \varrho_{\max})$.

The three cases defining $\widehat{r}(U,X)$ are separately illustrated in Fig. 3.5 and depend on the average rating $\overline{r(U)}$ of each person U:

- when U assigns the same rating to every song, then $r(U,X) = \overline{r(U)}$ for every song and $\widehat{r}(U,X)$ falls back to the constant 0 (solid line in Fig. 3.5);
- when U has an average rating of ϱ_{med}, then $\widehat{r}(U,X)$ falls back to a linear function (dashed line in Fig. 3.5);
- otherwise $\widehat{r}(U,X)$ is a logarithmic function, concave or convex whether the average rating $\overline{r(U)}$ is higher or lower than ϱ_{med} (dotted lines in Fig. 3.5).

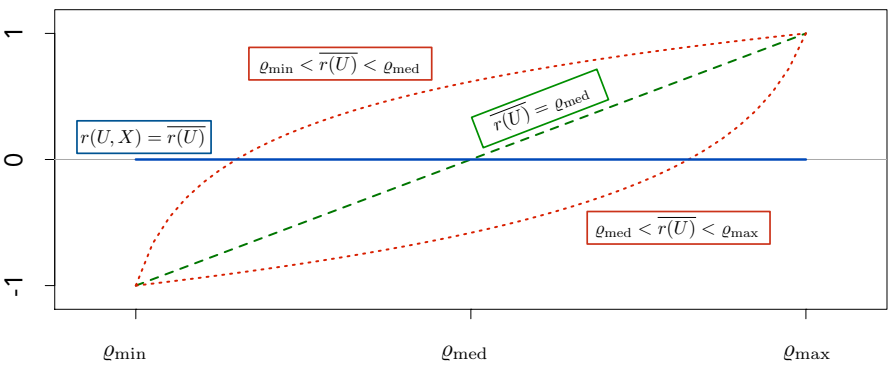

Figure 3.5: Normalised user ratings corresponding to different values of $\overline{r(U)}$.

Working with normalised values, every rating is interpreted with respect to each user, and the same absolute rating (e.g., 4 stars) can assume different normalised values $\widehat{r}(U,X)$ for different people.

Example 4. Figure 3.6 represents the situation where two friends U and V have both listened to ten songs $\mathcal{C} = \{X1, X2, \ldots, X10\}$ and have rated them with

different criteria. Even though U and V both assigned four stars to song $X5$, the normalised user rating for this song differs: $\widehat{r}(U, X5) = 0.67$ is positive, while $\widehat{r}(V, X5) = 0$ is not. The reason is that U assigns in average two stars to each song, so the observed value $r(U, X5) = 4$, higher than the average, corresponds to a positive normalised user rating, while V assigns in average four stars to every song, so the observed value $r(V, X5) = 4$ is not equally significant.

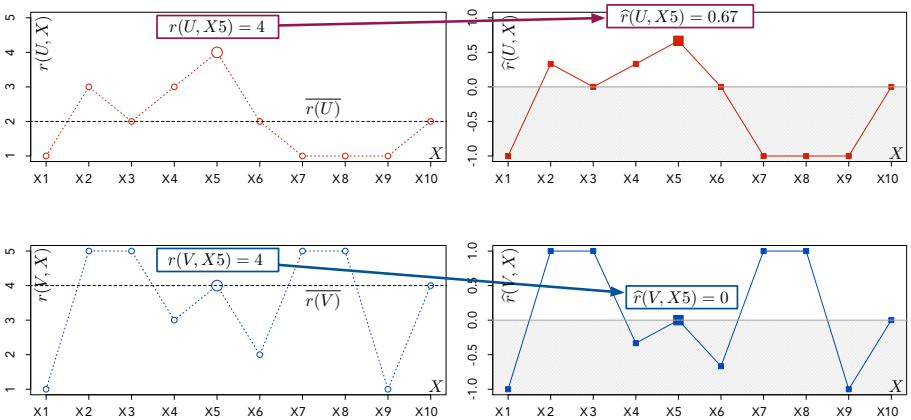

Figure 3.6: Effect of the normalisation process on user ratings.

3.4.3 Combining user ratings and play counts

Following the same approach described to normalise user ratings, play counts $n(U, X)$ can be normalised as well, resulting in a normalised play count function $\widehat{n} : \mathcal{U} \times \mathcal{C} \rightarrow [-1, 1]$ that yields positive (resp., negative, null) values for songs played more than (resp., less than, as much as) the average play count.

Normalising user ratings and play counts helps identify songs for which a person has a particular interest: every song X for which $\widehat{r}(U, X) > 0$ or $\widehat{n}(U, X) > 0$ has obtained by U a rating or play count higher than the average and, as such, can be assessed as one of the songs preferred by U.

Formally, the **individual preference** degree $i : \mathcal{U} \times \mathcal{C} \rightarrow [0, 1]$ that measures how much U likes each song $X \in \mathcal{C}$ is defined as a linear combination of these normalised properties:

$$i(U, X) = \pi \cdot \max(0, \widehat{r}(U, X)) + (1 - \pi) \cdot \max(0, \widehat{n}(U, X)) \tag{3.3}$$

where $\pi \in [0, 1]$ denotes the importance assigned to ratings with respect to play counts.

The reason why both normalised rating and play count are limited to a minimum value of zero is that implicit preferences are assessed only for songs for which either the rating or the play count is above the average. No 'negative' assumption is made about songs rated or played less than average. The reason

is that a person may have not rated or played a specific song for several reasons (lack of time, of a digital player, of the right occasion) and this should not be always interpreted as a lack of interest. On the other hand, having taken the effort to rate or play often a song should be positively interpreted as an implicit measure of interest.

This technique allows to estimate individual music preferences without any explicit action by the listener, exploiting only knowledge implicit in the listening habits data of a person.

Example 5. The music library shown in Fig. 3.1 contains 8 rated songs, with an average rating of 3.5 stars; five songs were rated above this value. Similarly, 15 songs have been played, the average play count is 4 and four songs have a play count above this value. The implicit preference of the author for these songs can be calculated with the function (3.3) assigning the same importance to ratings and play counts ($\pi = 0.5$). The results are reported in Table 3.1: 'Interstate Love Song' (Stone Temple Pilots) stands out as the author's favourite song, followed by 'Across The Universe' (Verdena) and 'My Little Empire' (Manic Street Preachers). A song such as 'Last Kiss' (Pearl Jam) is not in the list because both its rating (3 stars) and its play count (3) fall below the average.

Table 3.1: Individual preferences assessed for songs in Fig. 3.1.

song X		rating $r(U,X)$	rating $\widehat{r}(U,X)$	play count $n(U,X)$	play count $\widehat{n}(U,X)$	preference $i(U,X)$
3	Interstate Love Song	4	0.28	20	1	**0.64**
13	Across The Universe	5	1	3	0	**0.50**
14	My Little Empire	5	1	2	0	**0.50**
8	Heart Of Gold	4	0.28	9	0.5	**0.39**
1	Space Oddity	4	0.28	5	0.12	**0.20**
2	Two Of Us	—	—	5	0.12	**0.12**

3.4.4 Integrating additional properties

The individual preference degree (3.3) combines only two observable properties: user ratings and play counts. Other observable properties can be easily integrated in the function if they are found to be significantly related to musical preferences. The skip count (number of times a song was skipped before its end), for instance, can be considered as an indicator of 'negative' preferences. This property can be included into (3.3) by first normalising its values to the range

$[-1, 1]$, then adding this value to the linear combination (3.3) that defines the value of $i(U, X)$. The only precaution is to invert negative preferences: songs skipped *less* than the average would yield a positive normalised value and vice versa.

3.4.5 Extending to artists

Whatever the number of properties considered, songs that have never been 'experienced' (played, rated, skipped) are always assigned a value of $i(U, X) = 0$, which means that no implicit preference can be assessed.

This is an intrinsic limitation of the proposed technique: preferences can only be inferred for songs in the individual listening history. This behaviour can be improved considering that, in general terms, a person has similar preferences for songs by the same artist. For instance, if U has assigned 5 stars to ten songs by Björk, then U will probably like future released by Björk as well, since U shows to enjoy her music. In other words, a positive implicit preference for U can be assessed for *any* song by Björk, even those that U has not yet heard.

Formally, let X be a song that a person U has not yet experienced, and let $\mathcal{Y}_U(X)$ be other songs by the same artist of X that U has experienced:

$$\mathcal{Y}_U(X) = \{Y \mid a(X) = a(Y) \wedge i(U, X) = 0\}\,. \tag{3.4}$$

Then, the individual preference of U for song X can be estimated as the average preference for known songs by the same artist:

$$i(U, X) = \sum_{Y \in \mathcal{Y}_U(X)} \frac{i(U, Y)}{\#(\mathcal{Y}_U(X))}\,.$$

This extension enables the definition of a degree of implicit preferences also for songs outside of a personal music library.

3.5 Summary

This chapter has presented a technique to build individual music profiles from the analysis of personal listening habits. Listening habits describe which songs a person has been listening to and can either be extracted from personal music libraries or collected from Web communities (Last.fm, MusicStrands).

Of all the songs that a person has been playing, the preferred ones are those that have been played more often and have been best rated. This chapter has presented a technique to automatically measure how much a person likes each listened song. The proposed method analyses the play counts and ratings assigned by any person U to any song X and delivers a measure $i(U, X) \in [-1, 1]$ of musical preference.

The technique presented in this chapter will be employed in the poolcasting approach introduced in Chap. 4 to identify which songs satisfy most of the preferences of a given audience.

Chapter 4

The poolcasting technique

With this method that I have found
I'm redressing all I know

Mansun, 1998

4.1 Adapting music for a group of listeners

The first part of the thesis has explained how to use experience data from the Web to perform two specific tasks: estimate musical association between songs (Chap. 2) and estimate individual music preferences (Chap. 3).

Musical associations (which songs or artists sound well together in sequence) and individual preferences (what the public would like to hear) serve as the basis for the technique described in this chapter, which represents the main contribution of this dissertation. This technique helps identify, in a large repository of music, a subset of songs adapt for a group of listeners.

While disc jockeys (DJs) can use their senses to perceive the taste of the audience (watching their reaction, listening to their requests) and know from experience which songs mix well one after the other, here the methods introduced in the first part of the thesis are used to deliver group-customised music sequences (see Fig. 4.1).

4.1.1 Problem Definition

There are several situations where groups gather to listen to music, such as home parties, discos and, in a virtual sense, radio stations. In these contexts, someone is entitled to decide *which* songs to play; this can either be a professional DJ or someone from the audience.

In discos and AM/FM radios, expert DJs are appointed to select the best music for the audience. In home-parties, bars and online radios, this task is instead typically left in the hands of an automated system, for instance a portable music device (Apple iPod) with the 'shuffle' option turned on.

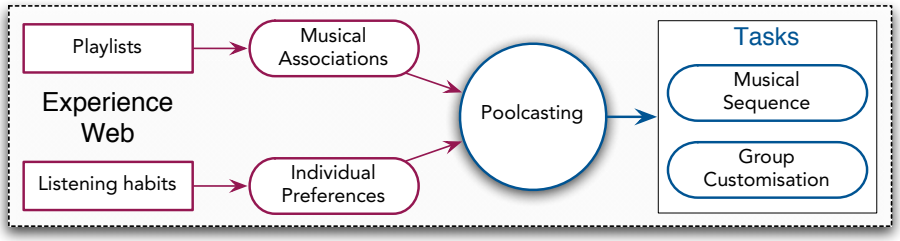

Figure 4.1: Collecting music experiences from the Web to deliver sequences of songs customised for groups of listeners.

The problem with **automated music selection** is that the preferences and reactions of the audience are not considered, so the result can be unsatisfactory for the actual listeners. Moreover, an automated selection does not ensure that each played song 'sounds well' after the previous one.

The problem addressed in this chapter is whether an intelligent technique can 'act as a good DJ', autonomously adapting the music to the listening audience. Precisely, the goal is to automatically select and deliver a musical sequence that matches four requirements:

Goal 1. **Variety**: no song or artist should be repeated within a short period of time;

Goal 2. **Smoothness**: songs should follow a sequence perceived as musically smooth;

Goal 3. **Customisation**: songs should match the musical interests of the current audience; and

Goal 4. **Fairness**: in the long run, every listener should have a similar degree of satisfaction with respect to the music played.

The first two properties are meant to build a good sequence of songs, avoiding repetitions (Goal 1) and ensuring a certain musical continuity (Goal 2). The last two properties are meant to customise the music for the group: selecting songs according to the musical taste of the audience (Goal 3) and embracing the preferences of every listener in the long run (Goal 4).

4.1.2 The poolcasting approach

This chapter presents poolcasting, an intelligent technique that addresses the music selection problem described above. Poolcasting generates musical sequences in real time, selecting at each moment which songs to play next based on the current audience and on the set of available songs.

The way in which the four goals of of variety, smoothness, customisation and fairness are addressed is by means of a Case-Based Reasoning (CBR) process that identifies which song to play at each moment.

CBR systems, given a new problem to solve, first *retrieve* similar past problems, then *reuse* their solutions to generate a good solution for the new problem, finally *revise* the proposed solution. Similarly poolcasting, to select at a given moment which song to play on a channel:

1. *(Retrieve Process)* first retrieves a subset of available songs (the retrieved set) that have not been played recently (Goal 1) and form a smooth musical transition with the last song played (Goal 2);

2. *(Reuse Process)* then ranks the retrieved set according to the preferences of the current audience (Goal 3) giving more importance to those listeners less satisfied with the music recently played (Goal 4); and

3. *(Revise Process)* finally considers the listeners' feedback to adjust individual preferences over time.

In order to know which songs form smooth musical transitions, poolcasting integrates the co-occurrence analysis process introduced in Chap. 2. In order to know the preferences of the current audience, poolcasting integrates the implicit user modelling approach described in Chap. 3.

The rest of the chapter is organised as follows. Section 4.2 reviews previous work related to Case-Based Reasoning, group-adaptive systems and social choice problems. The CBR process that combines experience knowledge from the Web to customise music for an audience is presented in Sect. 4.3. Section 4.4 explains how poolcasting can fairly satisfy the entire audience when listeners have different musical tastes.

4.2 Previous work

The approach described in this chapter reinterprets an artificial intelligence technique called Case-Based Reasoning (CBR) which is reviewed hereafter. Previous works on user-adaptive systems are then reported, with special focus on systems that adapt content to groups and have addressed the problem of customising music for an audience.

4.2.1 Case-Based Reasoning

Case-Based Reasoning [Kolodner, 1993; Aamodt and Plaza, 1994; López De Mántaras et al., 2005] is the process of problem solving based on the exploitation of past experiences, called cases, to propose solutions for present problems. Inspired by cognitive sciences, Case-Based Reasoning is based on the assumption that similar problems have similar solutions.

In CBR systems, knowledge is typically represented as a library of cases, also called case base. Each case holds knowledge related to a specific situation and is usually made of a (problem → solution) pair: the problem describes a task solved in the past and the solution explains how the task was carried out. New

tasks can be solved adapting past solutions, following a cycle that comprises four processes:

1. Retrieve: extract from the case base a subset of cases that present problems similar to the current one;

2. Reuse: adapt the solutions of the retrieved cases to the context of the current problem in order to generate a new solution;

3. Revise: evaluate and improve the outcome of applying the proposed solution to the current problem; and

4. Retain: store the newly generated (problem → solution) pair in the case base as a new case.

CBR has been applied in distinct domains where similar problems have similar solutions. In jurisprudence, for instance, the outcome of a lawsuit strongly depends on past court decisions; a valuable CBR system has been developed with a large case base of historical sentences that are retrieved and reused to help lawyers predict the verdict of new lawsuits [Weber-Lee et al., 1997].

Another area where CBR has proven helpful is medical diagnosis; CBR systems have been used to diagnose cancer [Díaz et al., 2006] and Alzheimer's disease [Marling and Whitehouse, 2001] comparing data and symptoms of each new patient (current problem) with data and symptoms of previous patients (past problems) who received specific cures (past solutions).

The minimal components of a Case-Based Reasoning system are the retrieve and the reuse steps, which generate a solution for a given problem. The main challenge in these steps is how to measure similarity between cases. The similarity measure depends on the problem description, ranging from a simple metric that corresponds to the distance between two features to more complex measures that depend on the application domain.

CBR techniques have also been employed in user-adaptive systems, delivering customised content to different targets based on previous interactions with the system.

4.2.2 User-adaptive techniques

User-adaptive techniques analyse the behaviour of their users, infer a model of their preferences and goals, and deliver content tailored upon their interests. These techniques have been applied to offer the right level of medical information according to the needs of individual patients [Hirst et al., 1997], to adjust educational presentations according to the expertise of the learners [Hohl et al., 1996], to give the right support to software users according to their usage [Horvitz et al., 1998], to tune the toughness of tracks in a car racing game according to the player's skill [Togelius et al., 2006].

User-adaptive systems have also appeared in literature labelled as adaptive interfaces, personalisation systems, adaptive hypermedia systems, user modelling systems, software agents or intelligent agents, and grow on the idea that it is worthwhile to learn something about each individual and adapt the response in some nontrivial way [Brusilovsky, 2001; Montaner et al., 2003].

Task: recommendation or delivery

User-adaptive systems can be distinguished according to their task being *recommendation* or *delivery*. A recommender system *suggests* some content of interest, while a delivery system *delivers* the actual customised content.

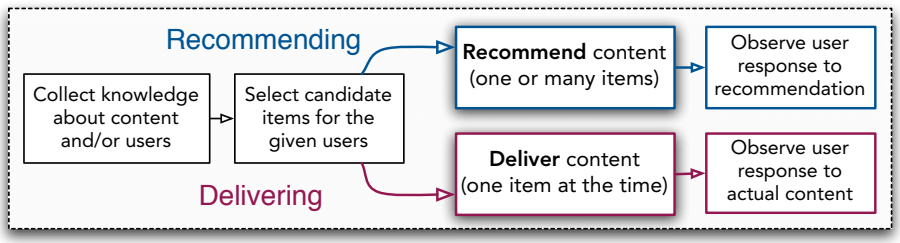

Figure 4.2: Recommender and delivery systems compared.

Recommender systems have gained wide attention in relation to the expansion of the World Wide Web, where they have proven helpful for people looking for new movies to watch [Hill et al., 1995], songs to listen to [Shardanand, 1994], news to read [Resnick et al., 1994], pages to visit [Lieberman, 1995], restaurants to check [McCarthy, 2002], people to know [Terveen and McDonald, 2005].

Delivery is appropriate when items are small, cheap, easily available and storable (e.g., digital audio, digital video, news stories) while items that are large, expensive or not easily available are only recommended (e.g., travel destinations, restaurants, social partners). Recommender systems can also work on multiple domains at the same time, like InterestMap [Liu and Maes, 2005] which simultaneously recommends books, songs, foods, films, television shows and sports based on cross-domain user taste profiles.

Recommender systems often offer unrequested recommendations; for example Amazon constantly presents lists of 'recommended books' that customers can completely (and often do) ignore. By contrast, delivery systems need be prompted by the users for customised content.

Since recommender systems do not provide the actual content, an immense collection of items can be recommended without the need for any physical storage. For example, MovieLens [Cosley et al., 2003] recommends movies from a catalogue of 5,600 titles; since movies are recommended — not delivered — the system only has to store and transmit the names of the movies, not the real movies. On the contrary, delivery systems usually require storage and transportation capacity which can limit the range of offer.

Output: items, sets or sequences

The outcome of an adaptive system can either be *one* item, a *set* of items, or an ordered *sequence* of items. One item is typical returned when items are expensive (e.g., travels, houses), and the audience can only afford to experience one. An example is CATS [McCarthy et al., 2006], which recommends *one* skiing travel destination for a group of friends.

A set of items is commonly returned when items require some time to be experienced (e.g., movies, books). For example, MovieLens recommends a *set* of movies to watch and Amazon recommends a *set* of books to read, leaving the users to choose which items to buy and in which order.

Finally, a sequence of items is returned when items require a short time to be experienced (e.g., songs, news stories), and the order in which they are experienced is significant. For example, Pandora delivers user-adapted Web radio channels with a specific 'musical continuity', in the sense that every song is acoustically correlated with the previous song played on the same channel.

4.2.3 Recommender techniques

Adaptive systems provide users with content of interest from a potentially overwhelming set of choices. Adaptation processes are characterised by different kinds of knowledge representation. A common family of techniques for adaptation is *collaborative filtering*, which consists in presenting a user with content liked by people with a similar profile. Another family of techniques is called *content-based filtering* and takes place by presenting a user with content similar to items previously liked by the same user. A third family of techniques, called *knowledge-based*, uses domain knowledge about items' association.

Collaborative filtering

Collaborative filtering techniques select items based on the correlation between people with similar preferences. The name "collaborative filtering" was proposed by the developers of Tapestry [Goldberg et al., 1992] and has been used ever since. The way in which similarity between users is computed depends on the application domain: Amazon interprets similar users as customers who bought the same books in the past, MovieLens as users who rated the some movies likewise. Once a similarity measure has been defined, the technique works by selecting for each user a set of items liked by people similar to that user. An interesting comparison of several music-related collaborative filtering systems (Last.fm, Pandora, GhanniMusic, Jango, MeeMix) is presented by Fox, 2007 [Fox, 2007].

Collaborative filtering is content-agnostic in the sense that the technique is suitable independently of the application domain (e.g., books, music, video). Herlocker et al., 2004 [Herlocker et al., 2004] discussed different problems of this approach related to the fact that items cannot be recommended unless users have expressed a preference for them. *Cold start* refers to the situation where

new items appear and the technique cannot immediately recommend them, since no user rating is available to work on. Similarly, when new users join in, the technique needs to learn their preferences before any recommendation can be given. *Scarcity* refers to situations where the number of available items is so much larger than the number of users that the coverage of ratings is very sparse and only a few objects are recommendable. Another typical problem is for users with *boundary taste* to be provided with poor recommendations since their profiles do not match many other user profiles. *Lack of diversity* refers to situations where a collaborative filtering technique suggests a set of items that are intrinsically similar (e.g., five books by the same author, five travels with the same destination) reducing the range of recommendations to a very specific target.

Content-based filtering

Content-based filtering techniques analyse the intrinsic content of every available item and recommend items which are similar to items the user likes. The way in which similarity between items is computed depends on the nature of items. InformationFinder assessed similarity between Web documents comparing the presence of significant phrases [Krulwich and Burkey, 1996], while NewsWeeder measured similarity between news stories comparing the frequency of the occurring words [Lang, 1995].

Examples of music content-based recommender systems are Mufin, which calculates the similarity between songs comparing properties like tempo, instruments, sound density or harmony [Mufin, 2008], The Echo Nest, which combines text analysis, audio analysis and user activity [The Echo Nest, 2009], and Tangerine, which works by analysing the BPM and the beat intensity of songs [Tangerine, 2009].

Content-based filtering is tightly related to the nature of the items to adapt and can obtain good results only in domains where content analysis, parsing and classification have advanced (e.g., text, music), while for complex domains where automatic analysis and categorisation is not yet feasible (e.g., video, interaction data), this is not a suitable approach, since similarities from item to item cannot be drawn as easily.

Knowledge-based techniques

Knowledge-based techniques reason about domain knowledge to find items that best match the user needs. These techniques do not rely on a base of user ratings and are independent of individual tastes. Instead they have knowledge about how a particular item meets a particular user need, and can therefore reason about the relationship between a need and a possible recommendation [Burke, 2000].

As Chun and Hong, 2001 [Chun and Hong, 2001] pointed out, knowledge-based techniques are appropriate for products like cars, houses, computers, that

a person would buy because of knowledge about how an item matches personal requirements rather than because other persons bought that same item.

A music-related knowledge-based system is the Music Genome Projects [Pandora Media, Inc., 2006]. As a group of trained music analysts listen every day to hundreds of songs and describe their musical qualities "using up to 400 distinct musical characteristics", a large knowledge base is built up of relationships between popular themes that serves as the basis for the recommendations provided through Pandora Web radio.

A sub-category of knowledge-based techniques is composed by systems that apply Case-Based Reasoning to the task of recommendation. In these systems, solutions to past problems constitute the knowledge required to provide recommendations for new users. Smyth, 2007 [Smyth, 2007] summarised advantages of case-based recommender systems, which can converse with the users to better focus on their requirements, evaluate critiques, enhance the diversity of the outcome and explain the reason behind the provided recommendations. Examples of Case-Based Reasoning recommender systems are Entree [Burke et al., 1997], the Wasabi Personal Shopper [Burke, 1999], Expertclerk [Shimazu, 2001], WEBSELL [Cunningham et al., 2001], Smart Radio [Hayes and Cunningham, 2001] and the CATS group recommender [McCarthy et al., 2006].

4.2.4 Group-adaptive systems

Adaptive systems can either be targeted to individuals or to groups. Extending adaptation processes to groups is a challenging problem, since each member has particular characteristics that should be accounted for. Messick and Brewer, 1983 [Messick and Brewer, 1983] pointed out how a group-adaptive system faces the "social dilemma" of looking for a compromise between individual and collective interests. The system has to find the behaviour that is adequate for the group while taking into account individual preferences of each member.

Group-adaptive systems have appeared in domains other than music, for instance McCarthy, 2002 [McCarthy, 2002] described a system that advices a group of friends with a restaurant to attend, O'Connor et al., 2001 [O'Connor et al., 2001] a movie to watch, McCarthy et al., 2006 [McCarthy et al., 2006] a tourist attraction to visit.

The way in which adaptation for the group takes place depends on the characteristic of the group. Hill et al., 1995 [Hill et al., 1995] distinguished between *co-located* and *displaced* groups. In the last case, the term "virtual community" is more appropriate: people can influence each other as though they interacted but they do not interact, and the system has to maintain displaced members aware of the actions and decisions taken by the group.

Haseman et al., 2002 [Haseman et al., 2002] distinguished between *changing* and *stable* groups, meaning that members are allowed or not to join and leave the group. If the composition of a group changes with time, a group-adaptive system might need to inform new members about how previous decisions were made.

Saklofske and Yackulic, 1989 [Saklofske and Yackulic, 1989] and Raghunathan and Corfman, 2008 [Raghunathan and Corfman, 2008] remarked how people in a group feel a sense of affiliation with the others and this can positively influence enjoyment from sharing a hedonic activity. While *intentional* groups take place when people explicitly long for a social experience to share with other people, *accidental* groups are made of people who gather with no explicit reason. McCarthy and Anagnost, 1998 [McCarthy and Anagnost, 1998] presented an example of how hard it is to satisfy an accidental group.

People in accidental groups are *competitive*, trying to push their individual goals in front of everyone else's, as in political elections. In intentional groups, people are instead more *collaborative*, trying to reach a certain degree of group satisfaction. Intentional groups are generally small groups for which sociologists have found that members may give up part of their individual goals to reach for some group effects, such as conformity, cohesiveness and "consensuality" [Hogg, 1996].

Chae and Flores, 1998 [Chae and Flores, 1998] used the term "narrowcasting" to describe the situation where media content is delivered to a small group. In a narrowcasting scenario, people are able to select from a long list of customised channels rather than passively experience a generic broadcast content. Narrowcasting enables the audience to precisely match the experience with their own views, although Chaffee and Metzger, 2001 [Chaffee and Metzger, 2001] pointed out how this confines people to live in a "cocoon of self-reinforcing media".

4.2.5 Preference aggregation

The main challenge for a group-adaptive system is how to combine preferences of multiple persons when they differ. There are distinct models to aggregate multiple individual preferences in order to assess the "suitability" [Jameson and Smyth, 2007] of a particular item for a group as a whole.

Preference aggregation is a multi-person decision making problem that has been broadly covered by social choice theorists [Arrow, 1970], social psychologists, economists and information scientists [Tindale et al., 2003; List and Pettit, 2002].

Different approaches proposed to aggregate group preferences include: introducing social functions that employ some pre-defined aggregation strategy [Masthoff, 2004]; asking users to provide information to determine how the combination can be accomplished [Jameson, 2004]; asking domain experts to guide the combination process [Ardissono et al., 2003]; promoting social interaction with an interface that allows people to watch, critique, suggest, and discuss recommended items [McCarthy et al., 2006]; deputing to agent negotiation, with automated agents acting on behalf of humans to generate and form group recommendations [Bekkerman et al., 2006]; modelling information markets through which group members bet on their judgement about future events [Sunstein, 2005]; developing genetic algorithms [Holland, 1975] to identify the item that best combines individual and group ratings [Chen et al., 2008].

The election of an aggregation function to merge individual preferences is conditioned by the required level of fairness. Depending on the application, various rationality principles may apply to select alternatives.

When aggregating multiple individual preferences into a unique group choice, a number of goals are desirable but not always compatible among them, such as matching the preference of the majority, preventing people from leaving the group, maximising average satisfaction, ensuring some degree of fairness, treating group members differently when appropriate, discouraging manipulation of the recommendation mechanism, ensuring comprehensibility and "acceptability" [Jameson and Smyth, 2007] and minimising misery. Misery refers to the situation where at least one member of the group is strongly unsatisfied by the aggregation made.

Chevaleyre et al., 2007 [Chevaleyre et al., 2007] illustrated different approaches to achieve a degree of fairness in the group, such as *egalitarianism* and *utilitarianism*. An egalitarian allocation is driven by the individual utility of the *poorest* agent in the system; aiming at maximising this value is an example for a basic fairness requirement. An utilitarian allocation is driven by the *sum* of the individual utilities; asking for maximal utilitarian social welfare is a very strong efficiency requirement.

The weakest possible requirement for preference aggregation is *Pareto efficiency* [Grabisch et al., 1998]: if at least one individual prefers an item X to an item Y and no one likes Y better than X, then the aggregated preference of the group should be higher for X than for Y.

Criteria of fairness change when the group-adaptive system does not select *one* item, but rather delivers a whole *sequence* of items, customised for the group. Behavioural decision theory [Novemsky and Dhar, 2005] suggests that the mere fact that an outcome is embedded in a sequence might create a frame of reference that can influence subsequent preferences. In these cases, the history of past aggregations should be taken into account to promote fairness.

4.3 The Case-Based Reasoning selection process

Poolcasting is a technique to adapt musical content for a group, and is focused around the idea of **channel**. A channel represents the virtual space where a group of people gather to listen to music together: a radio channel, a party location, a discotheque.

The persons listening to a particular channel are called the **audience** of the channel and are indicated with $\mathcal{U}$. People are free to join and abandon a channel at different moments: the audience of a channel changes with time.

The notation $\mathcal{U}_T \subseteq \mathcal{U}$ refers to the set of persons in the channel at a given time $T \in \mathbb{N}^+$ where time is measured counting the songs played so far on the channel: $\mathcal{U}_1$ denotes the audience of the channel while the first song is playing, $\mathcal{U}_2$ while the second song is playing, and so on.

The set of songs available to be played form the **content** of the channel and are indicated with $\mathcal{C}$. Similarly to the audience, content can change with time;

for instance newcomers can bring more CDs to a party or a radio can buy new records. The songs available at a particular time $T \in \mathbb{N}^+$ are indicated with $\mathcal{C}_T \subseteq \mathcal{C}$.

The problem addressed by poolcasting is to play a sequence of songs that matches the four goals of variety, smoothness, customisation and fairness introduced in Sect. 4.1. Since both the audience and the content change over time, the musical sequence cannot be entirely scheduled at time $T = 0$ but has to built in real time considering at each moment the available songs and current participants.

Poolcasting follows an **iterated** decision approach: while the first song is playing on the channel, poolcasting decides which will be the second song to play; as soon as the first song ends, the second song starts playing on the channel and poolcasting decides which song will play next, and so on. The way in which poolcasting determines which song to play at any given moment T is by means of a CBR process that is represented in Fig. 4.3 and whose components (case bases, retrieve, reuse, revise process) are explained hereafter.

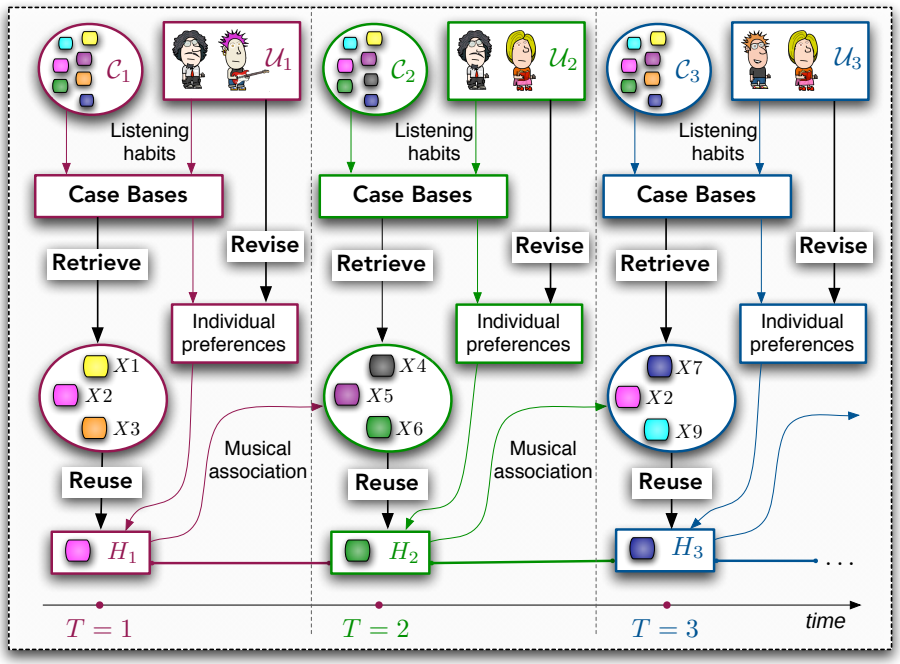

Figure 4.3: The iterative CBR selection process.

4.3.1 The case bases

In classical CBR, cases consist of (problem → solution) pairs; each new problem is solved independently by retrieving similar past cases and reusing their

solutions.

In poolcasting, the 'problem' consists in determining which song to play at each time T and cannot be solved *independently* from previous and successive problems. In fact the objective is not to find a *single* song that satisfies the audience, but to build a good *sequence* of songs that fulfils variety, smoothness, customisation and fairness.

For this reason, poolcasting does not store in the case base a series of 'past problems', but instead valuable knowledge to solve the sequential problem of deciding which song to play at any moment T.

Cases in poolcasting are defined as tuples ¡X, $a(X)$, $p(U, X, T)$¿, where $X \in \mathcal{C}_T$ is any available song, $a(X)$ its performing artist, and $p(U, X, T)$ is the **individual preference** of any listener $U \in \mathcal{U}_T$ for the song X, that is, how much U would like song X to be played on the channel at time T.

Poolcasting is able to estimate the individual preferences of the different participants following the method presented in Chap. 3. Every participant supposedly has a personal music library stored in some digital device (e.g., iPod, iTunes) and personal music libraries contain listening habits data indicating which songs a person has most played and voted. Poolcasting extracts the listening habits data of the audience either from their music players (e.g., from Apple iTunes, see Fig. 3.1) or from the Web (e.g., from Last.fm profiles, see Fig. 3.3) and infers the implicit preference $i(U, X)$ of each participant for each song, according to the function $i : \mathcal{U} \times \mathcal{C} \to [0, 1]$ defined in (3.3). These values are then stored in the case base as individual preferences.

If a participant V, for instance, has 'Karma Police' (Radiohead) as the top played/top rated track in the Last.fm profile page, then poolcasting stores in the case base the tuple ¡'Karma Police', Radiohead, 1¿ as a knowledge of the fact that V would like that song to be played. If a different participant Z has instead a slightly negative preference for 'Karma Police' (Radiohead), then poolcasting creates a second tuple ¡'Karma Police', Radiohead, -0.5¿ in the case base as a knowledge of the fact that Z would not like that song to be played.

If two different participants have different preferences for the same song (as in the previous example), poolcasting stores both facts since they can both help decide which music to play. To distinguish between cases related to different participants, poolcasting stores the cases of each participant in a separate case base. In other words, poolcasting holds a **collection of case bases**, one for each participant.

Each participant is treated independently and contributes with a piece of knowledge to the system. Since participants can join and leave the channel at different moments, the collection of case bases is dynamic: when new participants join the channel and share listening behaviour data, new case bases become available; when listeners abandon the channel, their case bases leave the collection. The collection of case bases is determined at each moment by the songs of the current participants in the channel.

4.3.2 The retrieve process

The retrieve process is meant to identify at time T which available songs are **good candidates** to be played next on the channel. The retrieve process addresses the goals of variety (Goal 1) and smoothness (Goal 2) with two subsequent steps: first each song is rated with a *relevance value* $k : \mathcal{C} \times \mathbb{N}^+ \rightarrow [0, 1]$ that expresses how much the song satisfies the requirements of variety and smoothness; then the κ best rated songs are retrieved.

Every song $X \in \mathcal{C}_T$ that does not fit the channel constraints is assigned the lowest possible value $k(X, T) = 0$ to prevent that song from being played. For instance, if a channel is defined as 'Rock' then every non-Rock song gets a relevance value of 0; if a channel is defined as 'Italian' then non-Italian tracks are assigned a value of $k(X, T) = 0$.

Every song recently played on the channel, or by a recently played artist, is also assigned a relevance value of 0. This is intended to guarantee the requirement of variety. Formally, let H_J be the J-th song played in the channel, and let $[H_1, H_2, \ldots, H_{T-1}]$ be the list of songs played so far. Then $k(X, T) = 0 \ \forall X \in [H_{T-\eta}, H_{T-\eta+1}, \ldots, H_{T-1}]$ (recently played songs) and $k(X, T) = 0 \ \forall X \,|\, a(X) \in [a(H_{T-\zeta}), a(H_{T-\zeta+1}), \ldots, a(H_{T-1})]$ (recently played artists). The parameters $\eta \in \mathbb{N}^+$ and $\zeta \in \mathbb{N}^+$ determine the number of songs that have to pass before the same song or artist can be played again in the same channel. The values of these parameters are defined according to the channel; ζ is small when the same artist is allowed to repeat (e.g., a 'Frank Sinatra' channel), while a high η characterises channels that strongly avoid repeating the same tracks (e.g. a 'Dance' channel).

Next, the retrieve process assigns a relevance value to the remaining songs according to how well they would go in a sequence after the last song played H_{T-1}. This is intended to guarantee the requirement of smoothness and is accomplished is by exploiting the co-occurrences analysis of playlists explained in Chap. 2. The idea is that the more people have played two songs together in their daily activities, the more two songs appear closely in playlists, the more they 'go well' together in sequence and the more it makes sense to reproduce them one after the other in a channel to guarantee smoothness. Formally, each remaining song X in the collection of case bases is assigned a relevance value of $k(X, T) = s(H_{T-1}, X)$ that indicates how much X would sound well in a sequence after the last song played H_{T-1}, where the function $s : \mathcal{C}^2 \rightarrow [0, 1]$ was defined in (2.8).

Having determined a relevance value $k(X, T)$ for each song $X \in \mathcal{C}_T$, the κ songs with the highest value are identified as the best candidate songs to become H_T (the next song to play). The cases that contain these songs are retrieved and constitute the *retrieved set*.

4.3.3 The reuse process

The reuse process is meant to adapt the retrieved set to the preferences of the current audience. In this process, the songs in the retrieved set are **ranked**

according to how much the current listeners like each song: the top ranked song is identified as the candidate preferred by the group of listeners 'as a whole'.

To know how much *the group* likes each candidate song, poolcasting reuses the individual preferences $p(U, X, T)$ stored in the retrieved cases. Every case contains knowledge about *one* participant. Poolcasting aggregates cases by candidate song to measure how much the whole group likes each candidate.

Several strategies exist to aggregate preferences in a group. A common strategy is *plurality voting*, which consists in ranking first the items preferred by the majority. With this approach, the preferences of the minorities never affect the determination of the best ranked item; in other words plurality voting does not endeavour to equally satisfy *all* the participants.

Poolcasting uses a different strategy which combines instead the preferences of all the members of the audience, assigning more importance to those participants that were less satisfied by the last songs played.

This strategy — called satisfaction-weighted aggregation and explained in Sect. 4.4 — returns for each candidate song X a group preference degree $g(X, T)$ that measures how much the group as a whole would like song X to be played at time T on the channel.

Given this group preference function, the reuse process ranks each song X in the retrieved set according to $g(X, T)$. The best ranked song is identified as the best candidate to become H_T, the song that will play next on the channel.

4.3.4 The revise process

So far, the CBR process has required no interaction from the participants. The retrieve process has automatically picked a subset of good candidate songs that have been ranked in the reuse process according to the preferences stored in the case bases.

At this point, poolcasting asks the members of the audience to express feedback about the proposed ranking. While some participants might agree with the top ranked candidate, others might prefer a different candidate song to be played. By reviewing their preferences for the candidates, participants can alter the ranking and make a different song play next.

The revise process consists in collecting explicit feedback from the current participants about the available songs. **Explicit feedback** is indicated with the function $e : \mathcal{U} \times \mathcal{C} \times \mathbb{N}^+ \to [-1, 1]$, where $e(U, X, T)$ denotes the last preference expressed by a participant U for a song X before time T.

The exact value of $e(U, X, T)$ depends on the type of feedback provided: $e(U, X, T) = 1$ denotes a participant U who stated 'unconditional love' for a song X; $e(U, X, T) = 0$ represents indifference; $e(U, X, T) = -1$ indicates that U does not want at all X to be played. Other values between -1 and 1 mean milder negative and positive preferences.

Explicit feedback serves to update the individual preferences $p(U, X, T)$ stored in the case bases. By default these preferences are inferred from personal listening habits, but if a participant explicitly states a feedback about a song, then the implicit preferences are replaced with explicit ones. Formally, the value

of the individual preference degree $p : \mathcal{U} \times \mathcal{C} \times \mathbb{N}^+ \to [-1, 1]$ is determined as follows:

$$p(U, X, T) = \begin{cases} e(U, X, T) & \text{if } e(U, X, T) \text{ is defined} \\ i(U, X) & \text{otherwise.} \end{cases} \tag{4.1}$$

As listeners state their preferences for the candidate songs, the ranking calculated by the reuse process is bound to change. For instance, if many listeners vote positively for a given candidate song, that song might become the new top ranked candidate. A similar outcome can occur if many participants vote negatively for the original top ranked song.

The revise process lasts for a finished period of time; when the time is over, the best ranked song is finally confirmed as *the* song H_T that will play next on the channel. This concludes the CBR process to select the T-th song to play, while the process to select the $T + 1$-th song starts right after.

4.4 The iterated social choice problem

The goal of poolcasting is to generate a musical sequence customised for a given audience. Poolcasting determines which songs to play iterating the CBR process: first, song H_1 is selected, then song H_2, and so on.

Each song belongs to a retrieved set of candidates that were not recently played and that go well after the last song played; this addresses the goals of variety (Goal 1) and smoothness (Goal 2). Moreover each played song is the top ranked according to the combined preferences of the audience; this addresses the goal of customisation (Goal 3).

The way in which multiple preferences are combined determines the degree in which fairness is addressed (Goal 4). The objective is to have every participant satisfied in the long run; therefore if some participants are not satisfied by the music played at a given moment, poolcasting should **reward** them by playing songs they like. In this way, everyone might listen to some of their favourite songs after a certain time.

Since the decision about which song is played is determined by the ranking of the candidate set, and since this ranking depends on the preferences of the audience, a way to reward unsatisfied participants is to aggregate preferences using a weighted average that assigns higher weights to less satisfied participants.

4.4.1 Group preference degree

The strategy to aggregate multiple individual preferences is called **satisfaction-weighted average**, and consists in averaging individual preferences with a weight that is inversely related to a participant's satisfaction.

Formally, let $q(U, T)$ indicate the individual satisfaction of a participant U at time T; the aggregated group preference is defined as:

$$\sum_{U \in \mathcal{U}_T} (1 - q(U, T - 1)) \cdot \frac{p(U, X, T)}{\#(\mathcal{U}_T)}. \tag{4.2}$$

This weighted average combines for each song X the individual preferences $p(U, X, T)$ of all the members of the audience $\mathcal{U}_T$, where the weight $(1 - q(U, T - 1))$ is inversely related to the satisfaction of each person. The function $q(U, T)$ that measures individual satisfaction is formalised hereafter.

4.4.2 Individual satisfaction degree

The individual satisfaction $q : \mathcal{U} \times \mathbb{N}^+ \to \{0, 1\}$ is defined as a combination of individual preferences for the songs played so far in the channel. In this combination, recent songs are assigned a higher importance since human satisfaction is an emotion that decays with time: people tend to forget remote experiences and be mostly affected by recent ones [Masthoff and Gatt, 2006]. The measure

$$\sum_{J=0}^{T-1} \chi^J p(U, H_{T-J}, T - J) \tag{4.3}$$

formalises this decay model: the individual preferences [$p(U, H_1, 1)$, $p(U, H_2, 2)$, ...] of a listener U for the songs played on the channel are averaged with a weight χ^J directly related to their recency, where $\chi \in [0, 1]$ is a parameter that determines the degree in which the recency of an experience influences its impact on satisfaction. Note that when $\chi = 0$ only the last song played has an impact[1] while when $\chi = 1$ every song has equal impact.

There are three issues to consider about (4.3). The first is that the formula combines the preferences of a participant towards *all* the songs played on the channel, from H_1 to H_{T-1}. Nevertheless, every participant is free to join and leave the channel at different moments, and only the preferences for the songs played *while U was listening* should determine the satisfaction of U. For this reason, a **channel membership** function $m : \mathcal{U} \times \mathbb{N}^+ \to \{0, 1\}$ is introduced to identify current listeners at time T:

$$m(U, T) = \begin{cases} 1 & \text{if } U \text{ is a member of the channel audience at time } T \\ 0 & \text{otherwise} \end{cases}$$

and is integrated into (4.3) to yield the formula:

$$\sum_{J=0}^{T-1} \chi^J p(U, H_{T-J}, T - J) m(U, T - J) \tag{4.4}$$

that aggregates only the preferences of U for the songs played on the channel while U was listening.

The second issue is that (4.4) yields values in the interval $[-(1 - \chi)^{-1}, (1 - \chi)^{-1}]$, while the desired satisfaction degree $q(U, T)$ yields values in $[0, 1]$. For this purpose, first a normalised version $h : \mathcal{U} \times \mathbb{N}^+ \to [0, 1]$ of the individual preference for the played songs is defined:

$$h(U, T) = \frac{1}{2} \left(p(U, H_T, T) + 1 \right) m(U, T)$$

[1] When $\chi = 0$, the first term of the sum contains 0^0, which is defined as 1 in this context.

where the more U liked song H_T, the closer to 1 the value of $h(U,T)$, while $h(U,T)=0$ if U was not listening to the channel while H_T was playing. Next (4.4) is rewritten integrating this notation, as follows:

$$\sum_{J=0}^{T-1} \chi^J h(U,T-J)$$

and finally this measure is divided by the sum of its weights in order to yield values in $[0,1]$:

$$\frac{\sum_{J=0}^{T-1} \chi^J h(U,T-J)}{\sum_{J=0}^{T-1} \chi^J m(U,T-J)}. \tag{4.5}$$

The third issue is that (4.5) is undefined for any participant that has not yet listened to any song, since the denominator equals zero. To solve this problem, a default **initial satisfaction** degree $\iota \in [0,1]$ is introduced that models how satisfied a participant possibly is before listening to any song. This value is integrated into (4.5) and the formula rewritten as:

$$\frac{\sum_{J=0}^{T-1} \chi^J h(U,T-J) + \chi^T \iota}{\sum_{J=0}^{T-1} \chi^J m(U,T-J) + \chi^T}.$$

This notation can be further reduced introducing the conventions that $m(U,0)=0$ and $h(U,0)=\iota$. Replacing in the numerator $\chi^T\iota$ with $\chi^T h(U,0)$ and, in the denominator, χ^T with $\chi^T m(U,0)$ eventually leads to define the individual satisfaction degree $q : \mathcal{U} \times \mathbb{N}^+ \to \{0,1\}$ as:

$$q(U,T) = \frac{\sum_{J=0}^{T} \chi^J h(U,T-J)}{\sum_{J=0}^{T} \chi^J m(U,T-J)}. \tag{4.6}$$

Individual satisfaction increases whenever a song that a participant likes is played, while decreases after listening to a disliked song. The impact of the last song played over satisfaction is not totally clear in (4.6) but can be unfolded rewriting the function in a recursive form that defines $q(U,T)$ only from the previous satisfaction degree $q(U,T-1)$ and from the preference $h(U,T)$ for the last song played:

$$q(U,T) = q(U,T-1) + \frac{h(U,T) - q(U,T-1)m(U,T)}{\sum_{J=0}^{T} \chi^J m(U,T-J)}. \tag{4.7}$$

This notation shows how every new song contributes with a 'delta' to the satisfaction of a participant. The impact of this delta (the second addend of the formula) is mostly determined by the decay parameter $\chi \in [0,1]$. If χ is small, the impact is strong, which means that satisfaction can change rapidly over time. At the extreme, if $\chi = 0$, then $q(U,T) = h(U,T)$, which means the satisfaction of a participant is solely determined by the last song played. On the other hand, if χ is large, then the impact is weak and satisfaction can only change slowly over time.

Figure 4.4 illustrates this behaviour with an example: the solid line plots the preferences $h(U,T)$ of a person U for 25 songs played on the channel, the dotted line plots the satisfaction degree of U modelled using a small decay value ($\chi = 0.2$); the dashed line plots the same satisfaction function modelled using a large decay value ($\chi = 0.8$). The dotted line is sharper and responds almost immediately to the changes of $h(U,T)$, corresponding to a model of 'immediate satisfaction'. The dashed line is instead softer and more influenced by the whole history of $h(U,T)$, corresponding to a model of 'long-term satisfaction'.

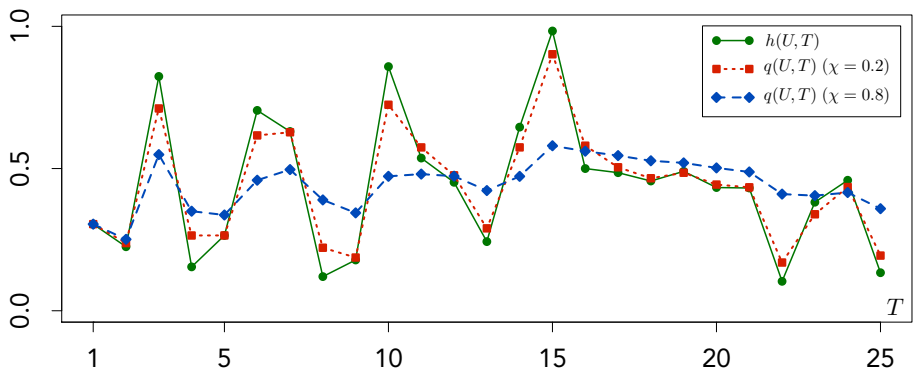

Figure 4.4: Influence of the decay parameter χ on individual satisfaction.

4.4.3 Satisfaction-weighted aggregation without misery

As defined in (4.2), the individual satisfaction degree (4.6) serves to weigh the influence of different participants when deciding which song to play next on the channel. This satisfaction-weighted strategy is intended to provide everyone (even participants with boundary musical tastes) with the chance to listen to some favourite songs.

One drawback of this strategy is that even songs that some participants 'detest' are bound to be selected. In other words, the fact that a participant has the lowest possible preference $p(U,X,T) = -1$ for a given song X does not prevent that song from being played if other participants with higher weights like it. As a result, *misery* can occur, which means that some persons are faced with items they do not wish to experience (songs they would not like to hear).

This situation can be avoided introducing a **misery threshold** $\mu \in [-1,0]$ that specifies the minimum individual preference that any member is willing to accept for a song. If any member of the audience has a preference for a song below this threshold, then that song is not played, independently of how much the remaining audience likes that song.

Integrating this threshold into (4.2) brings to complete the definition of the

group preference degree $g : \mathcal{C} \times \mathbb{N}^+ \rightarrow [-1, 1]$ as:

$$g(X,T) = \begin{cases} -1 & \text{if } \exists U \in \mathcal{U}_T : \\ & p(U,X,T) < \mu \\ \sum_{U \in \mathcal{U}_T} (1 - q(U, T-1)) \cdot \dfrac{p(U,X,T)}{\#(\mathcal{U}_T)} & \text{otherwise.} \end{cases}$$

To conclude, the strategy of poolcasting to address customisation (Goal 3) is to rank the retrieved set according to the group preference and to play the top ranked song. In order to achieve fairness (Goal 4), group preference is calculated with a weighted average that favours members that were less satisfied by the recent channel experience. Moreover, songs with very low individual preferences are avoided to prevent misery.

This preference aggregation strategy is iteratively used in the CBR selection process to generate a musical sequence made of songs that, over time, can satisfy the entire audience.

4.5 Summary

This chapter has described poolcasting, an automatic technique to adapt a sequence of songs to a specific audience. Most automated music selection techniques are not influenced by the actual listeners, while poolcasting delivers a sequences of songs customised for the audience. This is achieved by means of a CBR process, where the retrieve and reuse processes have been reinterpreted to generate a *sequence* of solutions adequate for a *group* of users.

The retrieve process does not look for *similar* cases but searches for *good candidates* for the sequence, employing the musical associations calculated in Chap. 2. The reuse process does not adapt *one* retrieved case but customises the retrieved set for the audience aggregating their music profiles assessed as described in Chap. 3.

Multiple individual preferences are combined with a satisfaction-weighted aggregation strategy that assigns different weights to different participants based on their satisfaction degree. The novelty of this aggregation strategy is that *memory* of past elections is accounted for in order to achieve fairness in the long run. The selection of each song depends on the previous songs selected and on their impact on listeners. The outcome is that individuals might occasionally be confronted with songs they do not like to keep the rest of a group happy, but are soon rewarded with some of their favourite songs.

The next chapter presents a real-world scenario where the poolcasting technique has been applied.

Chapter 5

Group-customised Web radio

It's that same sing song on the radio
It makes me sad

R.E.M., 1991

5.1 Adapting online radio channels to their audience

The previous chapter presented poolcasting, an intelligent music selection technique that automatically decides which music to play in contexts where people listen to music in group (discos, parties, radios, bars, offices).

This chapter focuses on the domain of online radios, presenting a novel Web radio framework that employs the poolcasting technique to customise radio music channels for their audience.

Most online radios cannot afford expert DJs to programme their channels and are scheduled with random sequences of songs of a given type (e.g., 'Rock' songs in a 'Rock' music channel). Employing an intelligent technique like poolcasting can significantly improve the quality of the music in terms of variety, smoothness, customisation and fairness.

Poolcasting Web radio autonomously schedules music channels with good successions of songs and enables listeners to interact among them and with the system in ways that are inconceivable in other Web radios. To name a few, listeners can:

- contribute with new songs to the music broadcast by the radio;
- define new radio channels that match their musical tastes; and
- evaluate the available songs to influence the music played.

The breakthrough of Poolcasting Web radio is the creation of a **social radio** experience. Listening to radio is normally an individual activity: people ignore who else is listening, cannot interact with each other or control music but for the volume knob. Most radio stations, both terrestrial and online, are not affected by *who* is actually connected and listeners are forced to waste time in quest of an 'ideal' channel.

Poolcasting Web radio takes instead inspiration from digital music services which broadcast music streams that are *personalised* for the audience. These services (e.g., Last.fm, Pandora) offer *private* music channels targeted to individuals, while Poolcasting Web radio provides *public* radio channels customised for multiple simultaneous listeners, enabling friends to enjoy music together. Poolcasting Web radio merges the distributed nature of online radios with the personalised nature of music recommenders.

5.2 Previous work

Poolcasting Web radio is a user-adaptive system that delivers customised music to a group. Previous music-related group-adaptive systems are reviewed hereafter, and a characterisation of Internet radios is provided.

5.2.1 Group-adaptive systems in music

A list of previous systems and prototypes designed to adapt music experience to a group of listeners is reviewed hereafter.

MusicFX [McCarthy and Anagnost, 1998] was a system that automatically selected which radio station to play in a gym according to the taste of the current attendants. The domain knowledge was made of a set of 91 radio stations categorised by musical style ('Album Rock', 'Beach Party', 'Flamenco', etc.); user profiles were built by hand, with each member specifying preference for each musical genre; the aggregation was computed using a weighted random selection policy, prohibiting the system from playing any station for which anyone present had a low rating and limiting the period of time that any one genre could play, in order to increase diversity. The main lesson learnt from MusicFX was that people had limited desire in compromising their musical preferences with those of some stranger with whom they just happened to work out with.

Smart Radio [Hayes and Cunningham, 2001] was a Web-based client-server application to share compilations of music. Smart Radio combined automated collaborative filtering and Case-Based Reasoning techniques. User ratings were gathered combining explicit and implicit feedback. Participants could explicitly choose to rate individual tracks or programmes, while the system implicitly gave a positive rating to songs that had been saved to a user's profile and to programmes built from scratch by a user.

Flytrap [Crossen et al., 2002] was an active environment that gathered users' musical tastes and automatically constructed a soundtrack to please everyone in the same room. The system combined domain knowledge about how genres of music interrelate and social knowledge about what kinds of transitions between songs people tend to make and broadcast music that satisfied user preferences while fitting its own notion of smoothness.

The Common Sense Disc Jockey [Ouko et al., 2002] was an application to automatically play music in a dance club environment. User profiles were built by a disc jockey observing the audience and completing a form with their age range, ethnic group, geographical provenience, political profile, while a camera tracker estimated the percentage of dancing people. Preference aggregation was computed following a "common sense" rule set; for instance if the crowd appeared to be young Asians, but the exact provenience and profession of the audience were unknown, the system concluded that cross-cultural music from international Pop singers like Britney Spears was best played.

Jukola [O'Hara et al., 2004] was an interactive MP3 juke-box that allowed active and collective participation in the choice of music in a public space. Jukola allowed people to nominate and vote for songs to get played by means of handheld devices wirelessly connected to a centralised juke-box. While a song was playing, each device presented four candidate songs to be played next drawn from the list of nominated songs, together with random choices. Each participant could cast a vote for a candidate and the best ranked song would be played next, according to a plurality voting strategy. Participants could also upload new songs from their devices to the juke-box to make more music available to the audience.

PublicDJ [Leitich and Toth, 2007] was a multiplayer game to propose and rate music to listen together with friends. At each round, users were requested to submit to a server songs they would have liked to hear. After analysing the received submissions, the server determined the best matching song with respect to the audience and the last song played. Associations between songs were discovered with an acoustic-based approach, combining low-level data (rhythm patterns, statistical spectrum descriptors, rhythm histogram) and high-level data extracted from the files. The system had no user models and did not keep memory of individual satisfactions.

PartyVote [Sprague et al., 2008] provided groups with a simple democratic mechanism for selecting and playing music at social events. Aggregated preferences were calculated with a plurality voting approach: songs frequently voted for and popular ones where more likely to be played, whereas boundary voters could listen to at least one song of their choicewithin two hours from their vote.

PartyStrands [Strands, Inc., 2006] was an interactive system developed by MusicStrands to have bar attendants decide which music would be playing through the night by means of text messages. Participants communicated their favourite artists with their mobile phones, and tracks by these artists were played from the venue's loudspeakers (see Fig. 5.1). The main problem for PartyStrands was a lack of context: every kind of music could be requested, resulting in Heavy Metal tracks immediately followed by Hip Hop tunes or Jazz themes. Moreover, requests came at the cost of a text message, which favoured participants willing to spend more money. User requests were simply declined if they could not be satisfied (e.g., when receiving many text messages at the same time) and preferences were aggregated favouring the majority, which often resulted in heavily mainstream musical sequences.

Figure 5.1: PartyStrands in action.

5.2.2 Internet radios

Music is the driving force of the recent transformation of Internet into a social medium. People turn to the Web not just to consume music, but to publish, discuss, share and discover music with friends [McGuire and Slater, 2005]. Indeed social context in music adaptive systems is "much more important than previously thought" [McEnnis and Cunningham, 2007]. This preamble explains the success of **digital music services** such as MySpace, Last.fm and MusicStrands that provide hundreds of millions of users with virtual communities where to interact with other music lovers.

Given this social demand, it is surprising to observe how online radios offer a service that is very close to old-fashioned AM/FM radios. Internet radios first appeared in 1994 [WXYC, 2004] with the intention to broadcast music over the net rather than over the air. Millions of online radio stations are available, but the *listening experience* for the audience is still very limited: listeners ignore who else is connected to each station, cannot interact with each other nor request songs or influence the music played.

This has not stopped online radios from growing, with a weekly radio

audience estimated in 33 million listeners in the United States [Rose and Lenski, 2008] and a tendency to rise: while listening to AM/FM radio declined four percent, listening to online radio increased 18 percent and free streaming of online music increased 37 percent [NPD Group, 2005].

The main competitors for online radios are Web-based music communities that broadcast personalised music streams to their members. Last.fm — the largest of these communities with 37.3 millions monthly unique visitors [Miller, 2009] — offers unlimited personalised channels for a monthly subscription of 3 euros.

The difference between online radios and digital music services such as Last.fm is that online radios broadcast a fixed amount of music stations that everyone can browse and join at any moment, while in Last.fm channels are private: each members gets a personalised and unique music stream that cannot be shared with other listeners.

5.3 The Web interface

The basic service provided by Poolcasting Web radio is online music streaming. Similarly to other Internet radios, visitors navigate with their Web browsers to the radio home-page (see Fig.5.2), select one channel from the list and start listening to a continuous sequence of songs.

Listening to music is very simple: users just have to click on the 'Listen now' button and an appropriate streaming media player (e.g., iTunes, VLC, Winamp) opens up playing the music broadcast from that radio channel. Another option is to navigate to the Web page of a particular channel (e.g., the 'Love the '90s' channel in Fig. 5.3) and start the integrated Adobe Flash player to listen to music directly in the browser.

Tuning into a channel is a concise and intuitive process, so people who just want to passively listen to music can do so. However, users who want to actively participate in the selection of the music played can do so swiftly as well with the different functions offered by Poolcasting Web radio.

5.3.1 Sharing personal music libraries

Internet radios typically have a large repository of songs stored on a server that some administrators continuously maintain and update to include new releases.

Poolcasting Web radio abandons this *centralised* storage model in favour of a *distributed* model where the collection of music is provided by the same listeners. The radio introduces the ability for listeners to share their personal music libraries, that is, to contribute with songs they own to the collection of music the radio can broadcast.

To share their music libraries and become active participants, users click on the 'Share your library' link and indicate the path to the local folder where their personal library is stored (see Fig. 5.4). This folder has to be accessible from the network via HTTP and the music library managed with Apple iTunes. After a

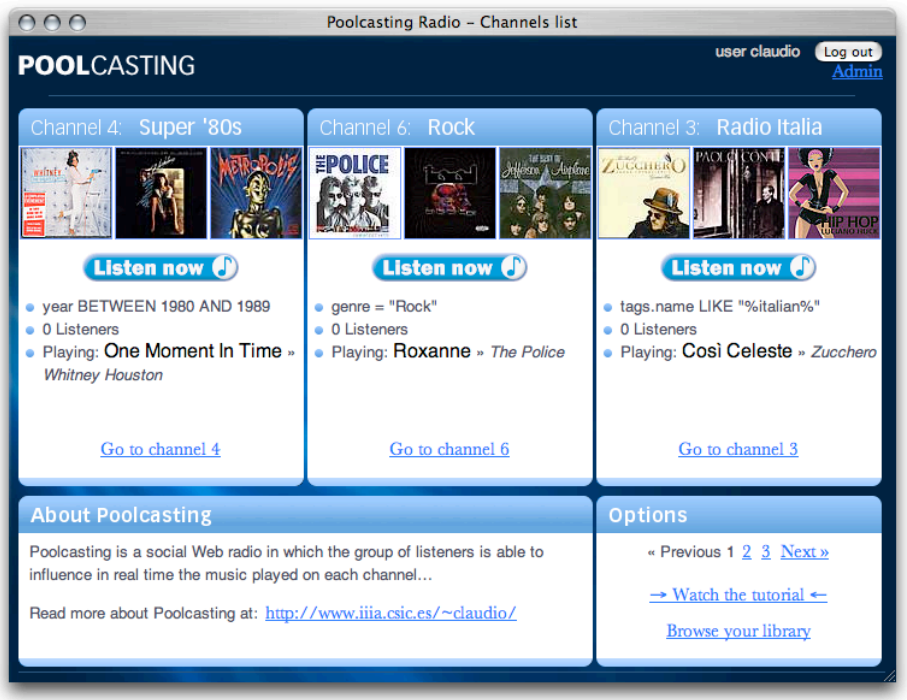

Figure 5.2: Channels list in Poolcasting Web radio home-page.

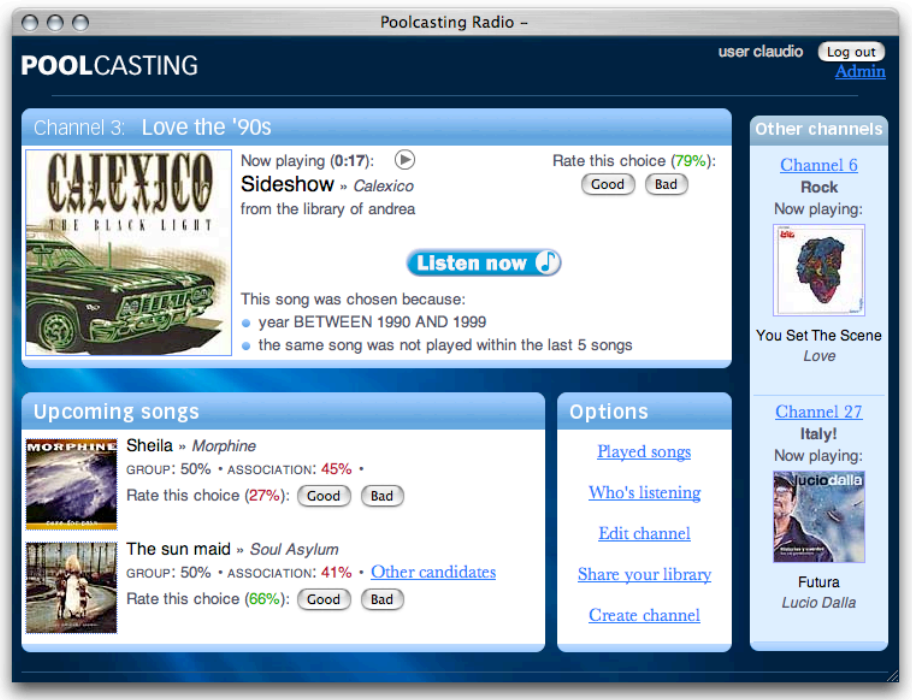

Figure 5.3: Web interface of a poolcasting radio channel.

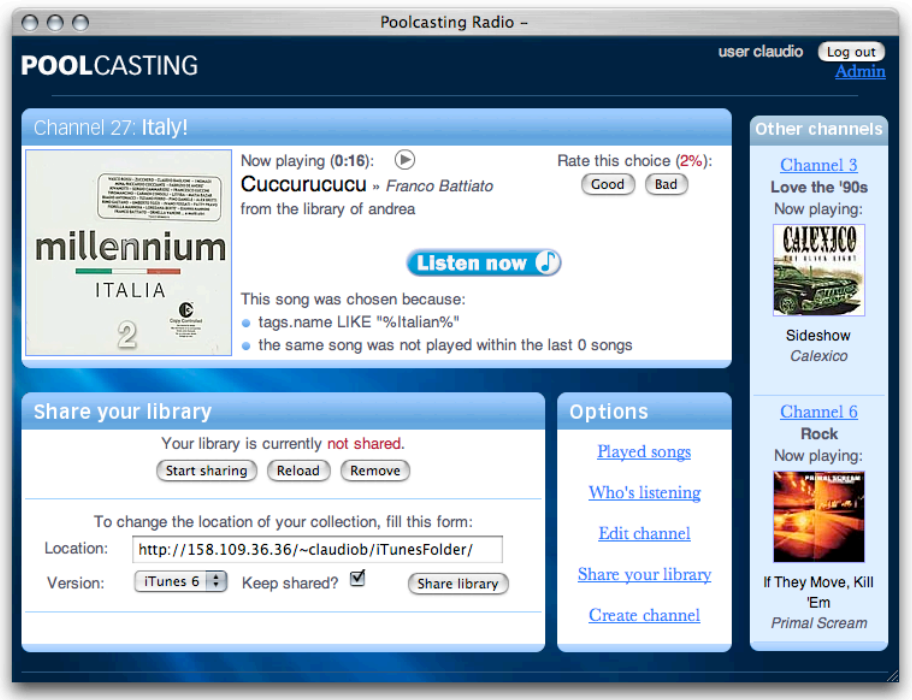

Figure 5.4: Sharing personal music libraries in Poolcasting Web radio.

listener has clicked on 'Share library', the system connects to the specified folder and uploads the 'iTunes Library.xml' file which contains the list of included songs and listening behaviour data.

Each song in the list is matched against two music recognition Web services. The first service is OpenStrands, a Web API provided by MusicStrands that returns a unique ID for more than six million songs, fixing wrongly spelled song titles or artist names (e.g., 'Obladi Oblada' instead of 'Ob-la-di Ob-la-da' or 'Mum' instead of 'Múm'). Additionally, each song is matched against the Last.fm database through the Last.fm Web API. This double check ensures a high identification rate and is also useful to retrieve further information such as the album cover, genre, year and tags of each song.

The 'iTunes Library.xml' file also contains the listening habits of each participant; as explained in Sect. 3.4 play counts and user ratings are used to automatically infer the preference $i(U, X)$ of each listener for each song.

Each shared library, together with the estimated preferences, is considered by the radio as a new case base. As described in Sect. 4.3, poolcasting holds a collection of case bases, one for each participant; each case refers to a song, its performing artist and its preference. Figure 5.5 illustrates with an example how two personal libraries are represented as case bases: songs are matched against OpenStrands Web service, their song and artist IDs are used to univocally identify each song X and artist $a(X)$ while the individual preferences $i(U, X)$ are assessed from the analysis of play counts and user ratings.

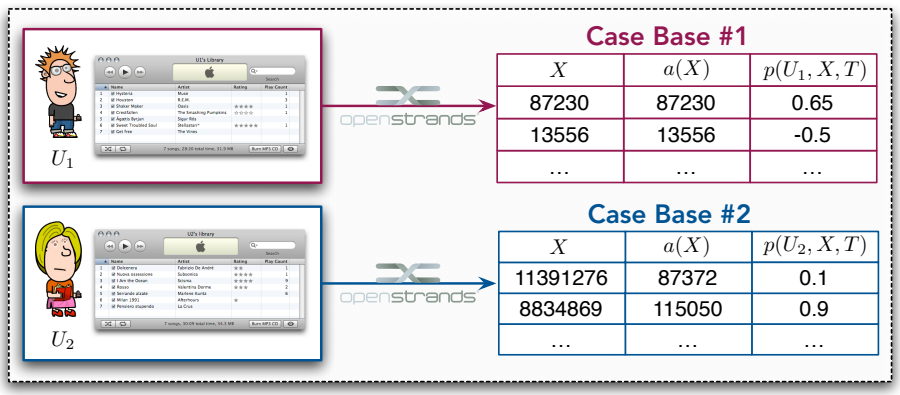

Figure 5.5: From personal libraries to case bases.

The last piece of information that is collected from the iTunes index file is the location of each song on the hard disk of the library owner. With this information, the radio can access any shared song as long as the corresponding library owner is connected to the network.

The collection of songs shared by all the listeners is called the **music pool**. There are several advantages of having a music pool distributed among multiple listeners rather than a centralised radio repository on a server. Firstly, a

distributed collection is more dynamic: when participants share their music libraries, the songs in their hard disks virtually enter the music pool; when one participant leaves, a portion of songs are removed; in general the music pool continuously varies over time. Next, a distributed collection is more up-to-date: while an administrator would usually update the radio library once a week or a month, individuals add new songs to their libraries every day and these instantly enter the music pool. A distributed collection also contains more of the longed for 'audio not available elsewhere': in fact personal libraries often include songs not publicly distributed (e.g., personal audio material, uncommon records, alternative versions of known themes). Finally, a distributed collection contains a finer selection of songs: while a centralised library is normally a massive heterogeneous collection of albums, not filtered by any quality criteria, personal libraries usually contain themes the owner has manually selected and implicitly appreciates.

The fact that the system retrieves and broadcasts music from personal libraries raises questions about copyright issues and licensing policies. While sharing and streaming music is not an illegal activity per se, most songs are currently published under restrictive licenses that prohibit free public diffusion. In order to avoid incurring in legal issues, the radio has two options. The first consists in limiting the service to songs published under non-restrictive licenses such as Creative Commons [Creative Commons, 2004] that do not forbid sharing or broadcasting. The second option is to pay the appropriate fee to the national collecting societies in charge of handing out the money to the corresponding music right holders. In Spain this monthly fee ranges between 56.26 and 450.11 euros depending on the nature of the service (commercial or not) and on the number of listeners [Sociedad General de Autores y Editores, 2006].

5.3.2 Creating radio channels

Online Web radios typically provide a large but fixed set of channels. Live365, for instance, offers about 6,000 radio stations, ranging from Christmas music to 80's Metal. Some of these may have no listeners at all; still the administrators will have spent time in their creation and money in songs acquisition, storage space and bandwidth. Large as the number of channels can be, some listeners may still be unsatisfied for they will not find *the* channel that best fits their taste.

Poolcasting Web radio abandons this authoritarian model and introduces the ability for listeners to create their own channels. Participants can browse the list of active channels and, if they do not find the one they like, create a new one by clicking on 'Create a channel' (see Fig. 5.6) and indicating its name, description and **channel pool**, that is, the subset of songs allowed to play. The channel pool is defined in terms of genres, tags and years, for instance "Jazz from 1967" or "Italian Electronic Dance". Every newly created channel automatically appears in the home-page and is programmed with music that fulfils the specified restrictions.

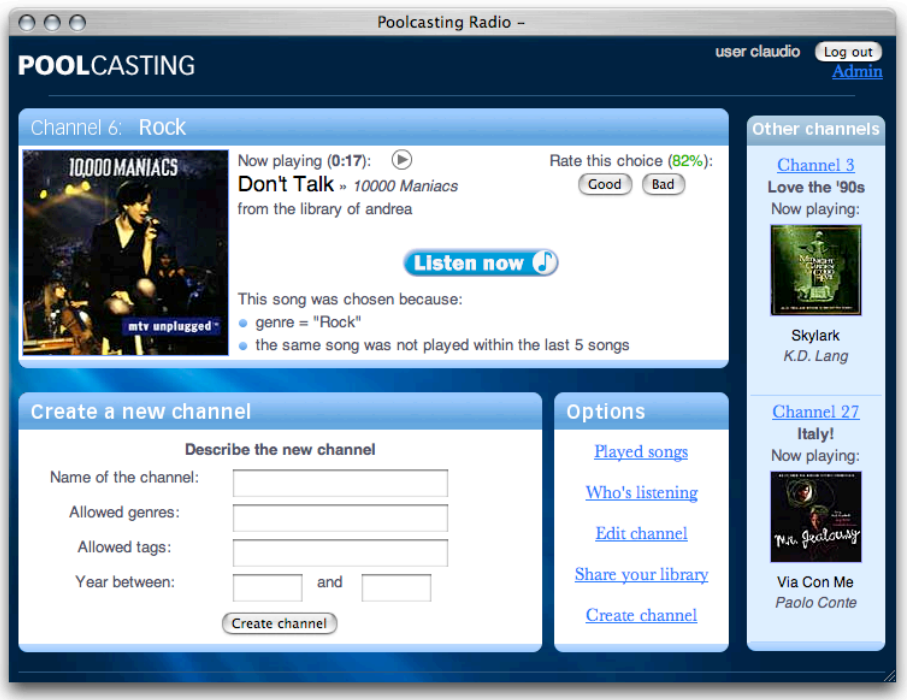

Figure 5.6: Creating radio channels in Poolcasting Web radio.

5.3.3 Expressing musical preferences

Poolcasting Web radio is similar to common radios in that songs cannot be skipped: they are played from the beginning to the end, one after the other in a continuous stream. The distinguishing feature is that listeners can express their preferences for every song in the radio and their preferences influence the music scheduled next on each channel.

Listeners can express their musical preferences in two ways. The first method is implicit: by sharing one's personal music library. Shared libraries are viewed by the system as case bases and the individual listening habits data contained (play counts and ratings) are automatically analysed to assess individual preferences $i(U, X)$.

The second method is explicit: providing feedback through the 'Good' and 'Bad' buttons that appear close to each song (see Fig. 5.3). The explicit feedback provided by a participant for a song is indicated by the function $e : \mathcal{U} \times \mathcal{C} \times \mathbb{N}^+ \rightarrow [-1, 1]$. The actual value of $e(U, X, T)$ depends on the most recent preference stated through the Web interface at time T, as follows:

- $e(U, X, T) = 0.5$ if U has clicked once on 'Good' for song X;
- $e(U, X, T) = 1$ if U has clicked twice or more on 'Good' for song X;
- $e(U, X, T) = -0.5$ if U has clicked once on 'Bad' for song X;
- $e(U, X, T) = -1$ if U has clicked twice or more on 'Bad' for song X;
- $e(U, X, T) = 0$ if U has first clicked on one button and then on the other one for song X.

As defined in (4.1), the **individual preference** $p : \mathcal{U} \times \mathcal{C} \times \mathbb{N}^+ \rightarrow [-1, 1]$ of a person for a song corresponds to the last feedback $e(U, X, T)$ provided, falling back to the preference $i(U, X)$ assessed by the listening habits if no feedback was ever provided.

5.3.4 Additional features

Another feature that characterises Poolcasting Web radio is the presence of a live chat integrated in every channel, which makes the members of the audience aware of who else is listening to the same channel (see Fig. 5.7).

Poolcasting Web radio also includes an administrative interface that enables a super user to control the various aspects of the radio, from starting and stopping channels to checking the available domain knowledge (musical associations and preferences), from listing the active participants to removing unused channels (see Fig. 5.8).

A feature that is common to AM/FM radios but is not present in Poolcasting Web radio is the chance for the audience to directly *request* specific songs. The reason is that Poolcasting Web radio longs to deliver a musical sequence with a certain musical continuity from song to song (Goal 2). Accepting direct

Figure 5.7: Discussing with other poolcasting listeners.

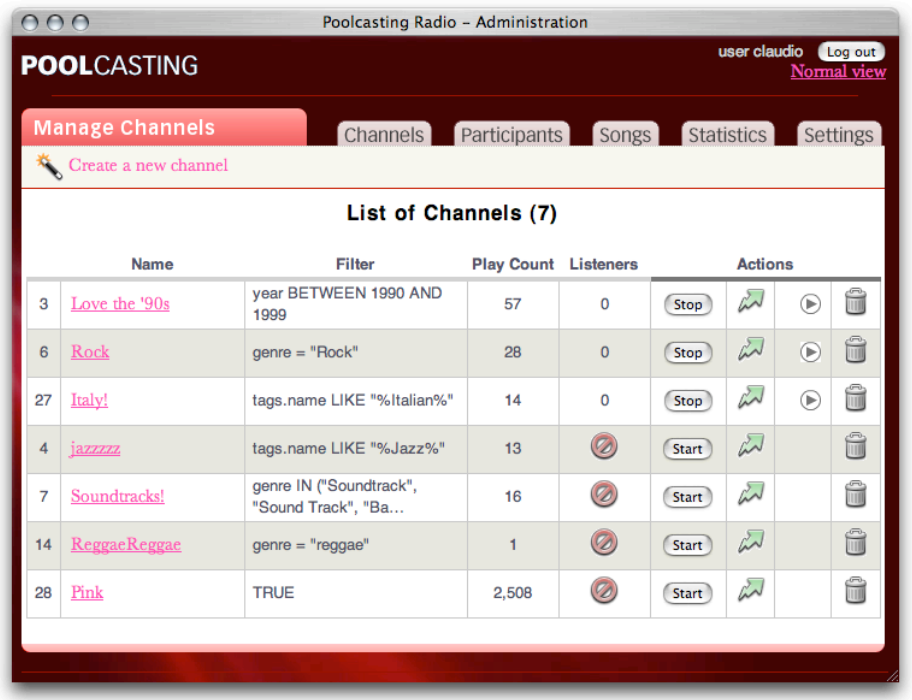

Figure 5.8: Administration panel of Poolcasting Web radio.

requests would force the radio to play any kind of music on each channel without guaranteeing smooth transitions from each song to the next. Direct music requests would also raise issues of fairness in case the system were faced with multiple simultaneous requests and would only be able to please one. Additionally, online radios that accept requests are regulated by more expensive licensing fees.

5.4 Automated music programming

The music that is played on each channel of the radio is automatically determined by an independent poolcasting CBR process. The whole radio is initially idle until the first participant joins the system and creates a new channel. At this point, the system spawns a poolcasting process that will determine in real time which songs will be played on that channel.

The songs available to be played are the songs shared by the current participants (music pool), although only a subset (channel pool) is allowed to play on each channel (e.g., only 'Rock' songs in a channel defined as 'genre = Rock'). From the channel pool, the system selects at random the first two songs H_1 and H_2. The radio connects via HTTP to the personal music libraries where these songs are stored, uploads them to the server and starts broadcasting the first song H_1 over the Internet. Then, the process selects which song H_3 to play next; this is accomplished by means of the CBR process introduced in Sect. 4.3.

First (retrieve process) the system identifies in the case bases a set of $\kappa = 15$ songs that have not been recently played and that form a smooth musical transition after H_2, the previous song in the queue. These constitute the retrieved set of songs that are good candidates to become H_3.

Next (reuse process) the system ranks the candidate set combining the preferences of the listeners with the satisfaction-weighted without misery strategy detailed in Sect. 4.4. The result is a ranked set where the top ranked candidate is the song that best addresses the goals of customisation and fairness according to the individual preferences stored in the case bases.

The top ranked song is shown in the channel Web page and, if participants do not review their preferences, is played on the channel right after H_2. However at this point (revise process) listeners have the chance to browse the list of candidates and express feedback for each of these songs.

Figure 5.9 illustrates the situation of a channel where the first song H_1 is playing ('Sideshow' by Calexico), the second song H_2 is on the server ready to be played ('Another Invented Disease' by Manic Street Preachers) and 'Infidelity (Only You)' (Skunk Anansie) is the best candidate to become H_3. By clicking on 'Other candidates', listeners can revise the whole retrieved set and update their preferences for every candidate. If a candidate receives a large positive feedback, its group preference degree can surpass that of 'Infidelity (Only You)' making that song become the new top ranked candidate.

The revise process continues until the currently playing song H_1 terminates. At that moment, the song H_2 starts being streamed, the current top ranked

Figure 5.9: Retrieved set for a Poolcasting Web radio channel.

candidate is determined as H_3 and uploaded to the server to be played next, and the CBR process starts again to select the song H_4.

As illustrated in Fig. 5.10, the music played on each channel is therefore automatically selected 'one song in advance'. While the first song H_1 is playing, the second song H_2 is uploaded to the server and poolcasting determines a set of good candidates to become H_3. The implicit preferences of the audience determine the best ranked song while participants can use the 'Good' and 'Bad' buttons in the Web interface to revise in real time these preferences. When H_1 ends, song H_2 starts, the best candidate to become H_3 is uploaded to the server and the process starts again to select H_4.

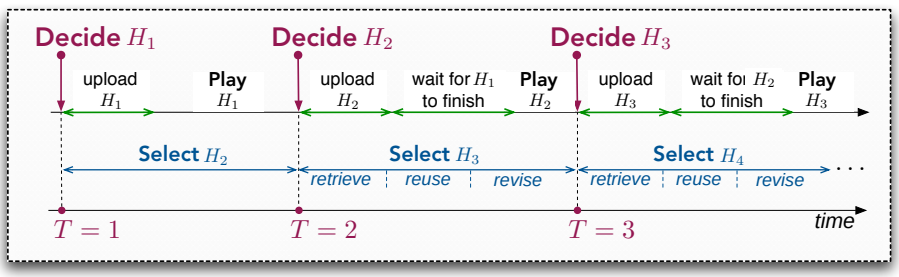

Figure 5.10: A song is playing, the next one uploaded and the next one selected.

The reason why poolcasting runs one song in advance is to guarantee an uninterrupted stream of music on each channel. Whenever a song is playing, the next song has to be already on the server to avoid gaps in the broadcast. Uploading a song from a personal library and encoding its audio into a proper streaming format require a certain amount of time. Running the process one song in advance provides the duration of an entire song (typically 3 to 4 minutes) to complete these tasks and also grants enough time to 'fall back' to another candidate is the top ranked one suddenly becomes unreachable (e.g., the library where it is contained disconnects from the network).

Music is broadcast from a centralised server rather than from personal libraries in order to guarantee a stable audio quality over time. If songs were streamed directly from personal libraries (e.g. with a distributed peer-to-peer architecture), the bandwidth of the broadcaster would represent a bottleneck for the audio quality of the stream. Moreover, if the broadcaster suddenly disconnected from the system, the stream would suffer an undesired break, which is avoided by streaming music from a centralised server after having uploading songs into a temporary buffer.

5.5 Implementation

The development of Poolcasting Web radio took about one year and its final architecture is illustrated in Fig. 5.11.

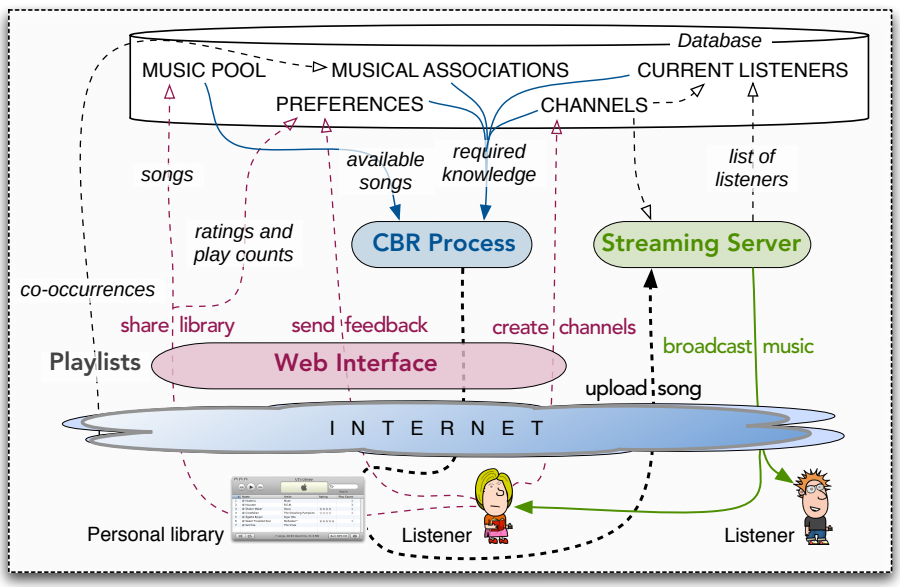

Figure 5.11: Architecture of Poolcasting Web radio.

The radio is idle until a participant creates a channel through the Web

interface. When this occurs, the streaming server opens a new Internet stream for the channel and a CBR process is started to fill the stream with music. The CBR process continuously checks in the database which songs are available (music pool), which ones fit the current channel (channel pool), how well each song would go after the last song played (musical associations) and how each of the current listeners like each song (individual preferences). Having gathered this required knowledge, the process returns the ranked set of candidates that participants can revise by sending 'Good'/'Bad' feedback through the Web interface.

When the best candidate is determined, the radio connects via HTTP to the library where the song is contained, uploads the audio file to the server and passes its content to the streaming server as an uncompressed audio signal. The streaming server encodes the audio into a proper streaming format (either MP3 or OggVorbis) that is later broadcast to the connected listeners.

The streaming server also maintains a list of the IP addresses of the listeners of each channel; channels that have not had any listener are automatically disabled after some time, in order to save bandwidth.

The entire framework has been running on a Mac Pro server equipped with two Dual-Core Intel Xeon CPUs at 2.66 GHz and 6 GB of memory. All the components of the radio have been developed with open source software: Ruby on Rails for the Web interface, Apache for the Web server, MySQL for the database, Ruby for the CBR process, liquidsoap [Baelde and Mimram, 2008] to manage the music queue and the stream generator, icecast for the streaming server.

5.6 Summary

This chapter has described a new Web radio framework that delivers group-customised music channels, offering an innovative social radio experience.

Poolcasting Web radio combines bottom-up and top-down approaches: users can express their musical preferences while the actual choice of music played is taken by a CBR process that iteratively checks which songs are available and who is listening and selects the songs that most satisfy the current audience.

The advantage of the radio architecture is that user interaction allows the system to model the musical preferences of each listener and exploit them to customise the content of the channels. Feedback expressed in terms of 'Good'/'Bad' preferences offers a direct overview of the listeners' preferences; in addition, the radio can work without any user interaction by exploiting the implicit knowledge contained in the personal music libraries shared by the participants.

Poolcasting Web radio merges the distributed nature of online radios with the personalised nature of music recommenders, thus offering a groundbreaking Internet service that can satisfy the needs of several Web music listeners.

Chapter 6

Experiments and evaluation

Everybody makes mistakes,
But I feel alright when I come undone

LCD Soundsystem, 2005

6.1 Working with a real Web radio

This chapter is divided in two parts. The first two sections describe the experience of ten actual users with Poolcasting Web radio: which channels they created, which songs they shared, which impressions they perceived. The rest of the chapter reports of experiments to evaluate how poolcasting can deliver radio channels that address the goals of customisation and fairness. A particular attention is dedicated to measure how the size and the musical homogeneity of the audience can affect the group satisfaction.

6.1.1 The active audience

An implementation of Poolcasting Web radio has been running for one year on a Web server located in the internal network of the Institut d'Investigació en Intel·ligència Artificial (IIIA-CSIC). The radio has been showcased in different research institutes, music-related companies and international conferences [Baccigalupo and Plaza, 2007b; Baccigalupo and Plaza, 2007c; Baccigalupo and Plaza, 2007d]; at each presentation the audience was asked to join and evaluate the system. A total of 29 persons accepted, ten of which have connected to the radio more than once. These 'active' users are male and 30 years old in average; five are Italian, four are Spanish and one is English.

6.1.2 The music pool

Eight of the ten participants agreed to share on the radio the songs contained in their personal music libraries. The total number of shared songs was

60,202. Matching these titles against OpenStrands and Last.fm Web services and discarding duplicates reduced the size of the music pool to 24,763 songs.

Five of the eight shared libraries were online 24 hours a day since they were stored on computers that were never switched off; the remaining three libraries were instead only available when the computers where they reside were connected to the Web.

The most common genres for the songs in the music pool were: Rock (3,499 songs), Soundtrack (2,670), Pop (1,609), Alternative & Punk (1,156), Electronic (614), Alternative Rock (560) and Indie (504). Most songs were published in 2005 (1,642 songs), while the average year of publication was 1999. Matching the music pool against Last.fm Web service revealed the following tags as the most common: rock & pop (9,138 songs), alternative (1,402), international (1,178), hard rock (529), r&b (520).

6.1.3 Listening habits

The listening habits data stored in the shared libraries revealed that most participants had never played or rated the songs contained in their libraries.

Only 4,564 of the 24,263 songs (18.4%) had ever been played by their owners in iTunes, while only 358 songs (1.4%) had been rated. The average play count for the 4,828 songs that had ever been played or rated was 1.8 times while the average rating was 3.3 stars.

These data interestingly point out how people may hold large music libraries in their hard disks and still play and rate only a small amount of songs, leaving most of their music collection untouched.

6.1.4 Observations

Although the radio was presented to about 100 persons, only ten became active participants. Informal interviews to those who did not join the radio revealed two main reasons. Some people were unfamiliar with online radios and were not interested in trying a new one. Others appreciated the idea of a social Web radio but did not have a music library managed with Apple iTunes, so could not actively participate.

The ten active users showed a noticeable interest for Poolcasting Web radio, to which they connected day after day. The number of shared songs was quite large compared to the number of listeners (in average 2,476 tracks per user), which enabled people to frequently discover unknown music.

A negative characteristic related to the active users was the low rate of played and rated songs in their personal libraries, which allowed poolcasting to build only partial models of their music profiles, restricted to a few songs and artists. The feedback provided through the ‘Good’ and ‘Bad’ buttons on the Web interface resulted very important to refine these music profiles over time.

6.2 Variety and smoothness

Fifteen distinct channels were created by the ten active users of the radio. Most channels were defined as single-genre (e.g., 'Rock' channel); others using periods (e.g. 'Music from the Eighties'); others combining genres and tags (e.g., a 'Soundtracks' channel playing tracks tagged as 'Soundtrack', 'Sound track', 'Banda Sonora' or 'BSO'[1]). Although participants could alter the definition of a channel after its creation, this function was barely used.

6.2.1 Variety

The variety of the music played on each channel (Goal 1) was achieved by setting the repetition parameters to $\eta = 50$ and $\zeta = 5$ (see Sect. 4.3): every song could be repeated on the same channel only after 50 other songs, while songs by the same artist had to be separated by at least 5 songs. These values were meant to avoid short-term repetition of the same music.

A problem reported by some listeners was that channels would sometimes repeat the same *sequence* of songs in consecutive days. For instance, whenever 'Romeo Had Juliette' (Lou Reed) played in the 'Rock' channel, it was always followed by 'The Wait' (The Pretenders).

The reason for this behaviour is that 'The Wait' is the top associated song with 'Romeo Had Juliette' in the music pool according to the knowledge in the data set of playlists. This means that, in the retrieve process, 'The Wait' always ranks as the best candidate to follow 'Romeo Had Juliette' and, in the reuse process, is the song selected to play next, unless the audience explicitly states a negative feedback.

To avoid this unpleasant behaviour, the requirement for variety was strengthened, forbidding that any *pair of consecutive songs* could be repeated two times in a row in the same channel. The song 'Romeo Had Juliette', for instance, would be followed by 'The Wait' (the top associated song) on its first appearance in the 'Rock' channel, while it would be followed by 'The Same Situation' (Joni Mitchell) — the second top associated track — on its second occurrence, and so on. This enabled the radio to achieve both short-term and long-term variety.

6.2.2 Smoothness

Poolcasting Web radio reuses the musical associations extracted from the analysis of playlists (see Chap. 2) to address the requirement of smoothness (Goal 2).

An example of smooth musical sequence delivered on the 'Rock' channel was: 'Pretty Noose' (Soundgarden), 'Bullet with Butterfly Wings' (Smashing Pumpkins), 'Kool Thing' (Sonic Youth), 'MFC' (Pearl Jam), 'Medication' (Queens of the Stone Age). This sequence shows a certain continuity since

[1] 'BSO' stands for 'Banda Sonora Original', 'Original Soundtrack' in Spanish.

all these songs revolve around the U.S. Indie Rock sound of the Nineties and go well together both for acoustic and cultural reasons.

Imposing smooth transitions from song to song comes at a price: the influence of the listeners is narrowed to decide only among a few candidates. As a result, a channel can take a long time to adapt content to the audience.

An example was reported by an active user of the 'Rock' channel who had to wait for the duration of four songs (about fifteen minutes) before listening to an Electronic Rock track, akin to his musical profile. The reason was that, because of the smoothness requirement, poolcasting could not 'jump' directly from one style to another but had to pass through a series of intermediate tracks before reaching an Electronic Rock track.

To avoid this behaviour, the smoothness requirement in Poolcasting Web radio was relaxed to include randomly chosen tracks in the retrieved set as well as songs that were musically related to the previous one. Specifically, the retrieve process was modified to retrieve the top 10 candidates from the list of associated tracks and the last 5 candidates at random from the channel pool. If the listeners ranked any of these songs better than the rest, that song would be played next, even without being the most associated with the last one. The Electronic Rock lover of the previous example, for instance, could in this way browse the retrieved set and pick the random candidate which was most akin to his musical taste, rapidly bending the music towards a more Electronic style.

With this approach, faster transitions could occur from a sub-genre to another when persons with different musical taste were present in a channel.

6.2.3 Observations

Other characteristics that the audience particularly appreciated or objected were revealed in a series of interviews to the radio listeners.

One participant found among the greatest qualities of Poolcasting Web radio "the idea of taking music out of personal libraries to new listeners". This was seen as a great way to get to know new composers and songs. Another quality was the "direct participation of the listener, which makes it a social radio". The main drawbacks observed were that "a channel can rapidly become overcrowded, negatively affecting individual satisfaction" and also the fact that musical preferences assessed from iTunes libraries can be wrong when the users are not "careful with their content, which can badly affect the quality of the transmission".

A different participant replied that the best aspects of the radio were "the interface, the streaming audio quality and the social aspect of the radio, even though I did not have the chance to fully delve into this". The worst aspect found was "having to wait for songs I did not like to finish". The participant also commented that "this kind of radio would work best in a shared office, where people are not just interacting *online* as they listen to the music".

A third participant rated positively the "comfortable interface", while was worried about the long time that had to pass before the radio could learn individual musical preferences. In his opinion, Poolcasting Web radio can make

listeners "quite satisfied" but not "totally satisfied", something that is positively "compensated by getting to discover new music that you are probably going to like".

A fourth participant appreciated the fact of being able to "rediscover songs from the bottom of my music library, that I did not remember having". Voida et al., 2006 [Voida et al., 2006] already investigated the satisfaction given by this "rediscovery" experience which is correlated to the increasingly large music libraries that characterise several iTunes users.

The main lesson learnt is that the listeners of Poolcasting Web radio implicitly accept a social compromise. When they join a channel, they are aware they will not possibly like every song played, yet they will have the chance to easily discover new music preferred by other listeners, with whom they can interact and discuss. This social component would not exist if they were to play music directly from their personal digital players or from other Internet radios.

6.3 Group customisation and fairness

The previous section reported on experiences of real users of Poolcasting Web radio and on the way in which radio channels achieve a level of variety (Goal 1) and smoothness (Goal 2) that is appropriate for the audience. The rest of the chapter evaluates whether the poolcasting technique can achieve the goals of customisation (Goal 3) and fairness (Goal 4) on groups of different sizes and musical homogeneity.

To obtain objective results, independent of the particular music profiles of the ten active users, the evaluation is run in a simulated environment with a set of artificially created profiles.

6.3.1 Random music profiles

The first experiment evaluates the performance of poolcasting in a radio channel with five participants. Each person is simulated by an artificial music profile that describes which songs a participant likes and dislikes. Precisely, five music profiles are generated *randomly*, that is, assigning a random number in $[-1, 1]$ to the individual preference $i(U, X)$ of each participant $U \in \mathcal{U}$ for each song $X \in \mathcal{C}$.

This is intended to emulate a sort of worst case scenario where five members of a channel do not share any particular musical affinity: one may like some Pop and Rock songs and detest Heavy Metal tracks, another may like Jazz and Rock and dislike Pop, and so on.

The five participants all join the same radio channel, defined with no filter: all the available music is allowed to play, from any genre, year or tag. The channel is started, and the first song is played at random from the available ones. All the successive songs are selected by the poolcasting CBR technique described in Sect. 4.3, with the parameters initially set to $\kappa = 15$ (size of the retrieved set), $\iota = 0.4$ (initial satisfaction), $\chi = 0.8$ (satisfaction decay) and $\mu = -0.75$ (misery threshold).

Whenever a new song is played, the values of two functions are registered for every participant U:

- $h(U,T)$, that is, how much U likes the song played; and
- $q(U,T)$, that is, how satisfied U is for the songs played so far.

After 25 songs, the channel is stopped and the registered values are analysed. In order to obtain significant results, the whole process (creating a channel, playing 25 songs, collecting results) is replicated 20 times with different initial random songs and music profiles and the analysis is performed on the average values of $h(U,T)$ and $q(U,T)$ through all the iterations.

This experiment is meant to evaluate whether poolcasting can fairly satisfy a group of five listeners characterised by random music profiles. This can be considered as a sort of worst case scenario since in the real world listeners of the same radio channel typically share some musical affinity.

6.3.2 Results

Figure 6.1 shows the result of this evaluation. Each line in the graph corresponds to one of the five participants. The plot on the left shows the individual preferences $h(U,T)$ of each participant for each of the 25 songs played; the plot on the right indicates their satisfaction degrees $q(U,T)$ after each song.

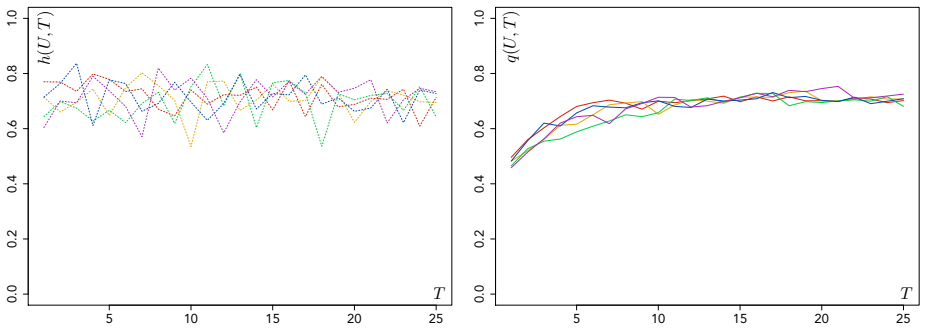

Figure 6.1: Individual preferences and satisfaction degrees of five participants listening to the same channel for the duration of 25 songs.

With regard to customisation, the results look positive. The whole audience likes every song, with an average individual preference of $\overline{h(U,T)} = 0.72$. Lower spikes in Fig. 6.1 (left) indicate songs that a participant did not particularly like; even in these occasions the preference $h(U,T)$ never falls below a value of 0.5. Moreover, lower spikes are immediately followed by high values, demonstrating the power of poolcasting in rewarding participants who did not like some of the recently played songs.

The impact of this balanced strategy over individual satisfactions is illustrated in Fig. 6.1 (right). All the lines are quite close and stable over

time, indicating that poolcasting achieves fairness by keeping every participant similarly satisfied in the long run.

The difference between the most and the least satisfied listener is small: $q(U,T)$ has an average standard deviation of 0.04 which means that a reasonable degree of fairness is achieved. The satisfaction of the group as a whole is quite high for this kind of worst case scenario, in fact the average satisfaction degree over the 25 songs played equals $\overline{q(U,T)} = 0.68$.

6.4 Discordant listeners

The previous section has evaluated poolcasting on a group of listeners with random individual preferences. In that case, poolcasting was able to deliver a group-customised sequence of songs to satisfy the entire audience.

This section considers a different situation in which five participants are split into two purely antagonist groups: the music that three users prefer, the other two dislike, and vice versa. No song exists that the entire audience likes.

The rationale of this experiment is to prove whether poolcasting can achieve fairness with discordant listeners, determining a sequence of songs that, in the long run, can match the preferences of all the participants. Again, this is a sort of worst case scenario: in the real world, listeners of the same radio channels typically share musical affinities for some songs.

For the sake of this experiment, five artificial profiles are created to simulate five participants split into two discordant groups. Three users are assigned with random *positive* preferences $i(U,X)$ for half of the available songs and with random *negative* preferences for the remaining half; the opposite occurs for the other two users.

Figure 6.2 (left) illustrates this distribution on a sample set of ten songs: the users labelled as 1, 2 and 3 only like the last five songs, while the users labelled as 4 and 5 only like the first five songs. By comparison, Fig. 6.2 (right) represents the scenario previously evaluated in Sect. 6.3 where individual preferences were independently assigned among listeners.

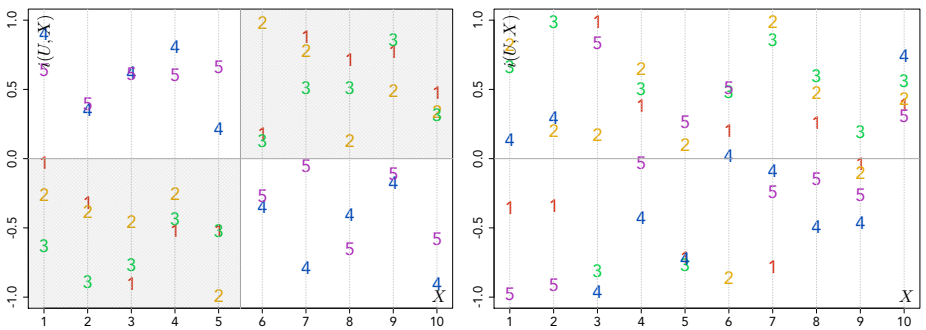

Figure 6.2: Sample individual preferences of five participants whose profiles belong to two discordant groups (left) or are independent (right).

The evaluation is run as in Sect. 6.3, having the channel playing 25 songs, repeating the process 20 times with different initial songs, and observing the values of $h(U,T)$ and $q(U,T)$ at each iteration.

6.4.1 Results

The results of the evaluation are shown in Fig. 6.3: the plot on the left marks the individual preferences $h(U,T)$ for the songs played; the plot on the right indicates the satisfaction degrees $q(U,T)$ of the five participants.

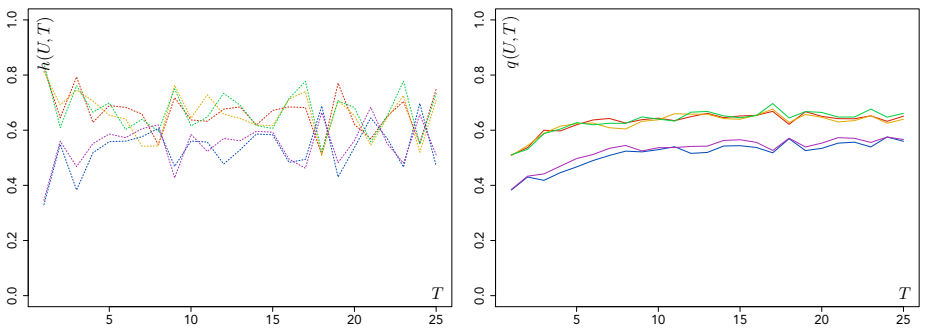

Figure 6.3: Individual preferences and satisfaction degrees of five participants split in two groups with discordant music profiles.

The first observation is that the two groups can be clearly identified in the graphs: most played songs are preferred by three participants who, as a consequence, have a higher satisfaction degree.

Still, the difference between the two groups is not very large and whenever a song preferred by the majority is played, a song preferred by the minority is played soon after. The songs played at position 8, 18, 21 and 24, for instance, are preferred by the minority more than by the majority. Moreover, every song played, although preferred by one of the two antagonist groups, has been selected to be reasonably acceptable by the members of the other group, albeit to a lesser degree.

The motivation for this balanced outcome is the satisfaction-weighted aggregation strategy presented in Sect. 4.4 that combines individual preferences rewarding less satisfied listeners. In this case, two participants are generally less satisfied than the other three so once in a while poolcasting delivers one of their favourite songs, to achieve fairness.

Thanks to this approach, the distance between the satisfaction degrees of the groups does not increase over time, as shown in Fig. 6.3 (right). This result is notably positive considering that actual radio channels do not typically have an audience split into two purely antagonist groups.

6.4.2 The importance of memory

In the previous experiment, a certain degree of fairness was achieved thanks to the satisfaction-weighted aggregation strategy that holds memory of previous outcomes to determine which items to deliver next.

To prove that this strategy is the responsible for the positive results obtained, an equivalent experiment is run under the same conditions but using a different aggregation strategy, with no memory of past decisions.

The experiment is run with the same audience of two discordant groups but preferences are here combined with a simpler approach, selecting at each step the song that has *in average* the highest preference, independently of the previous songs played. No memory of past satisfaction is kept.

The results of this experiment are shown in Fig. 6.4 which illustrates the average individual preferences (left) and satisfaction degrees (right) of the five participants for the songs played.

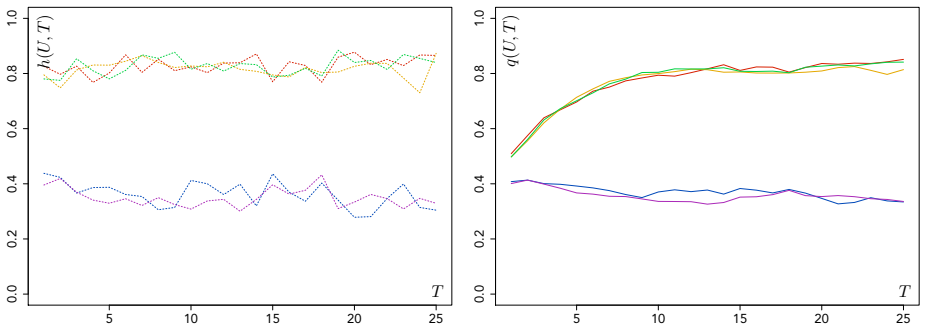

Figure 6.4: Individual preferences and satisfaction degrees of five discordant participants when songs are selected without memory.

As can be observed in Fig. 6.4 (left), in this case the radio channel plays only songs that three participants (the majority) prefer more than the remaining two (the minority). As a consequence, three members of the audience are more and more satisfied as time goes by, while the remaining two members are less and less satisfied, as shown in Fig. 6.4 (right).

The lesson learnt is that, without memory of past elections, fairness can only be achieved punctually, satisfying at each moment only the current majority. The satisfaction-weighted aggregation strategy that characterises poolcasting, on the other hand, can fairly satisfy the entire audience over time, playing songs that both the majority and the minority like to a certain extent.

6.5 Concordant listeners

The next experiment is meant to evaluate the performance of poolcasting in the case members of the audience share some affinity with respect to their music profiles. While the previous sections focused on worst-case scenarios made

of random or antagonist participants, this section considers a more common situation in which different listeners of the same radio channel like the same kind of music.

In this case, profiles are not generated independently one from the other, but with a certain concordance. Precisely, for every song $X \in \mathcal{C}$ in the music pool, the preferences $i(U, X)$ of the five participant are generated as random numbers limited to a particular sub-range of $[-1, 1]$ of diameter $\vartheta \in [0, 2]$, that is:

$$\max_{U \in \mathcal{U}} i(U, X) - \min_{U \in \mathcal{U}} i(U, X) \leqslant \vartheta \ \forall X \in \mathcal{C} \, .$$

The smaller the value of ϑ, the closer the preferences of different participants for the same song, the stronger the affinity between music profiles. This distribution of music profiles is illustrated in Fig. 6.5 on a sample set of ten songs.

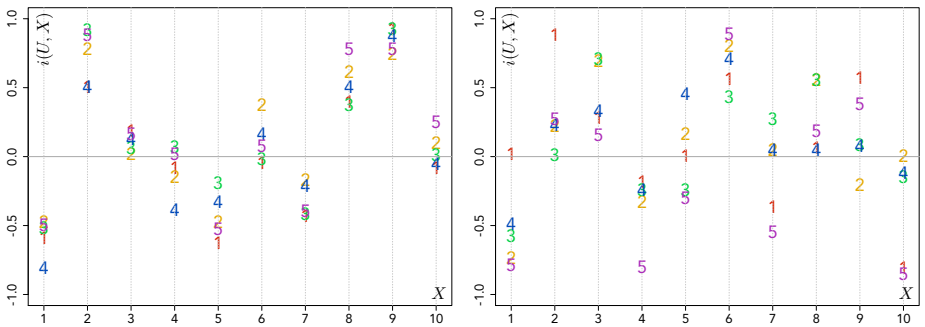

Figure 6.5: Sample individual preferences of five participants whose profiles share an affinity of $\vartheta = 0.5$ (left) or $\vartheta = 1$ (right).

Figure 6.5 (left) represents the situation where $\vartheta = 0.5$; in this case all the individual preferences $i(U, X)$ for the first song are restricted to negative values in $[-0.9, -0.4]$, all the preferences for the second song are positive with values in $[0.4, 0.9]$, all the preferences for the third song are medium with values in $[-0.2, 0.3]$, and so on. Figure 6.5 (right) illustrates the situation with a larger diameter $\vartheta = 1$; in this case the affinity is less evident. Note that when $\vartheta = 2$, the situation is identical to Fig. 6.2 (right).

The evaluation of the experiment is run as in Sect. 6.3, with five participants listening to 25 songs played on a channel, repeating the process 20 times with different initial songs, and observing the values of $h(U, T)$ and $q(U, T)$ at each iteration.

6.5.1 Results

Figure 6.6 (left) shows the results when $\vartheta = 0.5$, that is, when participants have a quite strong affinity in musical taste. Compared to Fig. 6.1 (right) (audience with no musical homogeneity), the results improve with respect to both the

average individual preference (increased to $\overline{h(U,T)} = 0.86$) and the variance among different users (the standard deviation decreases to 0.02).

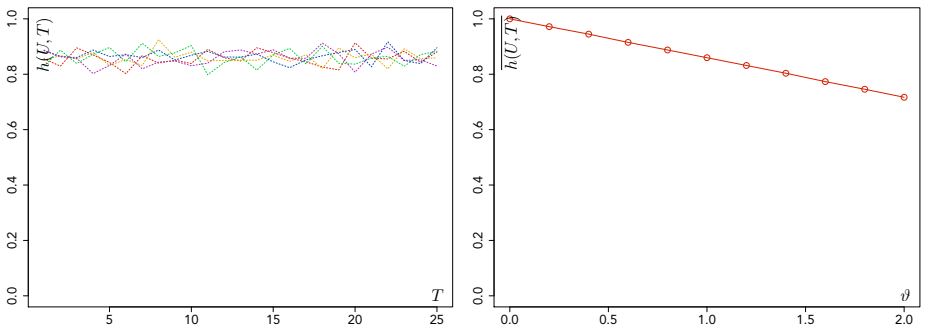

Figure 6.6: Individual preferences for songs played when $\vartheta = 0.5$ (left) and correlation between affinity degree and average preferences (right).

The relationship between group homogeneity and satisfaction is illustrated in Fig. 6.6 (right), which shows how poolcasting is better able to deliver songs the audience strongly likes when the value of ϑ is small. In the real world, a group of people listening to the same radio channel is expected to present some homogeneity in their musical taste, in which case the performance of poolcasting improves noticeably.

6.6 Scalability

The next experiment evaluates the impact of the size of the audience over the group satisfaction. The previous sections have only considered channels with five listeners; this section investigates whether adapting content for less or more people can affect the performance of poolcasting.

The evaluation is run using two new sets of artificial listeners: the first contains only two user profiles while the second simulates a group of twenty people. As in Sect.6.3, profiles are generated assigning random values in $[-1, 1]$ to the individual preference $i(U, X)$ of each person for each song. A channel is run for the duration of 25 songs, the process repeated 20 times and the satisfaction degrees $q(U, T)$ are analysed.

6.6.1 Results

Figure 6.7 shows the satisfaction degrees of the participants in a group of size 2 (left) and size 20 (right).

Compared to the group of size 5 represented in Fig. 6.1 (right), the satisfaction degrees are higher in Fig. 6.7 (left) and lower in Fig. 6.7 (right). In other words, poolcasting has it easier to adapt content for smaller groups than for larger groups. This makes sense since the more the listeners, the more

the music preferences to fulfil, and the harder to satisfy the entire audience. Specifically, the average satisfaction degree decreases from $\overline{q(U,T)} = 0.84$ with size 2, to $\overline{q(U,T)} = 0.72$ with size 5, to $\overline{q(U,T)} = 0.61$ with size 20.

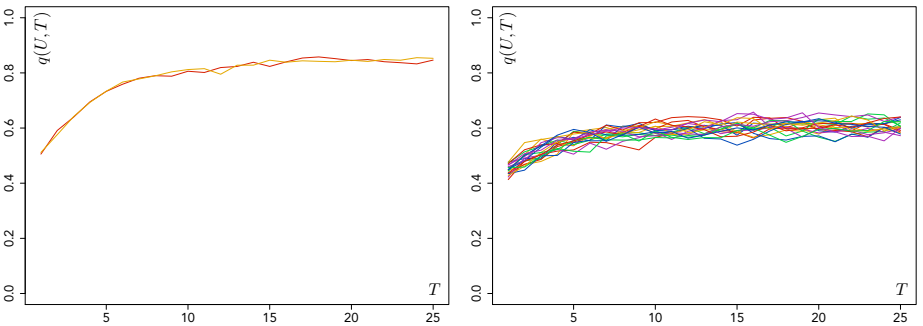

Figure 6.7: Satisfaction degrees for a channel with 2 participants (left) or 20 listeners (right).

More interesting, though, is that the variance between participants remains small even when the size of the group grows. The lines plotted in Fig. 6.7 (right) are all very close and stable over time, meaning that poolcasting is able to keep a good balance of the satisfactions of all the listeners in the long run.

Although one might conclude from this experiment that poolcasting can only be applied to radio channels with a small audience, this conclusion is not correct for a series of reasons.

Firstly, this section has focused only on a worst case scenario where all the listeners have random music preferences. In the real world, members of the same channel typically share some musical affinity, which makes it easier to find songs that satisfy multiple listeners at the same time. As a consequence, group satisfaction is not always as badly affected by the group size as illustrated in Fig. 6.7 (right).

Secondly, the experiment has been run on a channel with no filters, playing songs from every genre, tag or period. However most online radio stations restrict the music played to a specific set of songs (e.g. a 'Rock' channel) to attract a public with a certain homogeneity in musical taste. As observed in Sect. 6.5, the stronger the affinity in the music profiles of the participants, the higher the satisfaction degree that poolcasting can provide, even for large groups.

Lastly, populated channels can benefit the listeners in terms of music discovery. A listener U connected to a channel with a large audience is more exposed to unknown music, played because of the preferences of other participants. Even though unknown songs will not match the music profiles of U, causing a decrease in the satisfaction degree $q(U,T)$, the chance of discovering new music is seen as a positive trade off by many listeners.

The advantage of poolcasting is that, even when the audience is very large, fairness is achieved keeping the satisfaction of all the listeners balanced in the long run.

6.7 Other parameters

The experiments in this section evaluate the impact of three other parameters of the poolcasting process over customisation and fairness.

The first parameter is the retrieval size κ, which determines how many songs are selected in the retrieve process as 'good candidates' to be played next.

The second parameter is the misery threshold μ, which determines the minimum individual preference that any listener is willing to accept for any played song.

The third parameter is the initial satisfaction ι, which determines the default value of $q(U,T)$ for a participant who has not yet listened to any song in the channel.

All the experiments are performed under the conditions described in Sect. 6.3, with five artificial listeners with random music profiles.

6.7.1 Retrieval size

The retrieval size κ, introduced in Sect. 4.3, indicates how many songs are selected in the retrieve step of the CBR process. The larger the retrieval size κ, the more the candidate songs that are pre-selected to be played next, the higher the probability to find a candidate song the audience will like.

Figure 6.1 (left) illustrated the outcome on individual preferences of running poolcasting with a default retrieval size of $\kappa = 15$. Figure 6.8 (left) evaluates the same process but with the parameter set to $\kappa = 30$.

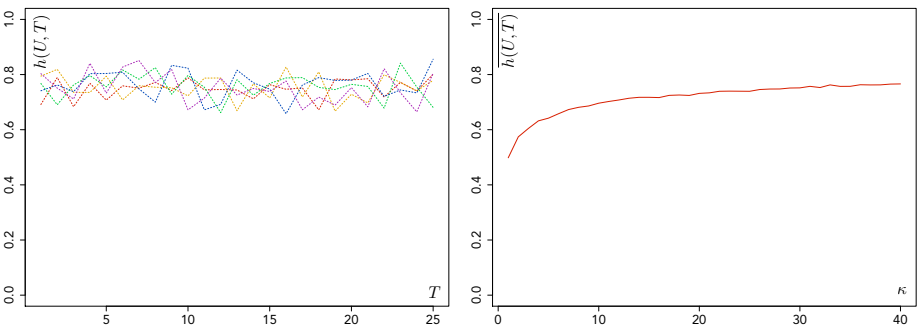

Figure 6.8: Individual preferences for songs played when $\kappa = 30$ (left) and correlation between retrieval size and average preferences (right).

Having doubled the retrieval size, the average group preference increases but not noticeably, from $\overline{h(U,T)} = 0.72$ (with $\kappa = 15$) to $\overline{h(U,T)} = 0.76$ (with $\kappa = 30$). Figure 6.8 (right) confirms this observation, showing the relationship between the retrieval size κ and the average group preference $\overline{h(U,T)}$. A clear improvement occurs in the lower range of intervals, but for values of κ larger than 15 the average group preference remains quite stable.

For this reason, $\kappa = 15$ can be considered as a good retrieval size. A smaller value would worsen the quality of the candidate set and, as a consequence, the quality of the played songs. A larger value, on the other hand, would not improve much the quality and could decrease the performance of poolcasting, required to rank more candidate songs in real time.

Another good reason not to have a large retrieved set is that Poolcasting Web radio offers a Web page where listeners can send feedback for the candidate songs (see Fig. 5.9). If the list were too long, the audience would probably just ignore it, rather than stating a feedback which is very valuable to refine individual music profiles.

6.7.2 Misery

Another parameter that affects the music selection process is the misery threshold $\mu \in [-1, 1]$, introduced in Sect. 4.4.3. This value specifies the minimum preference degree that the audience is disposed to accept for a played song. For instance, if $\mu = 0$ then no song that any listener dislikes is allowed to play on the channel. At the extreme, when $\mu = -1$ every song can be played; when $\mu = 1$ only songs that the whole public unconditionally loves can be played.

If the misery threshold is low, then every song can be played, even songs one participant detests (*low minimum* individual preference) and everybody else loves (*high group* preference). If the misery threshold is high, then only songs that no one detests (*high minimum* individual preference) can be played, even if they have a *low group* preference.

These two cases are illustrated in Fig. 6.9, which represents the preference degrees of listeners for the songs played when the misery threshold is low ($\mu = -1$) or when the value is higher ($\mu = 0$). The higher the threshold, the lower the average preference degree: $\overline{h(U,T)} = 0.73$ with $\mu = -1$ (left), $\overline{h(U,T)} = 0.59$ with $\mu = 0$ (right), while $\overline{h(U,T)} = 0.72$ with $\mu = -0.75$, as shown in Fig. 6.1 (left).

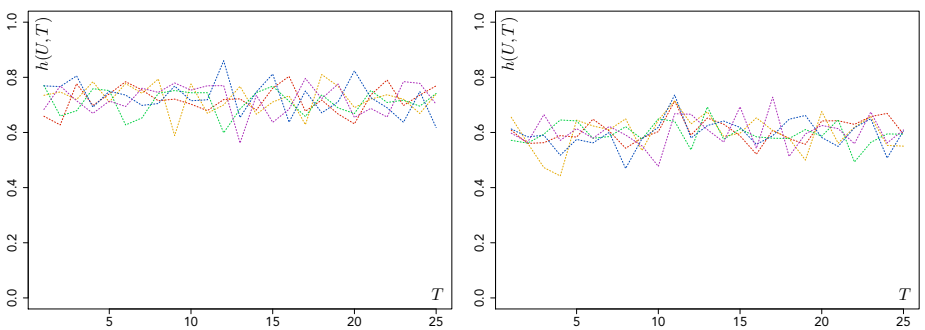

Figure 6.9: Individual preferences for songs played when misery threshold $\mu = -1$ (left) or when $\mu = 0$ (right).

The lesson learnt is that the misery threshold is better kept small, since large

values force poolcasting to select songs that will not bother anyone but that no one will really love either.

The threshold could as well be set to $\mu = -1$, to completely ignore the issue of misery. In fact, poolcasting listeners that do not like the songs played are already rewarded by the satisfaction-weighted aggregation. By ignoring the issue of misery, poolcasting would make it impossible for malevolent participants to take control over the music played by strategically casting negative feedback for every song except those they would like to hear.

6.7.3 Initial satisfaction

The last parameter that can affect the outcome of poolcasting is the initial satisfaction $\iota \in [0, 1]$, introduced in Sect. 4.4.2. This value defines the satisfaction degree of any person who has not yet listened to any song.

Figure 6.10 shows the effect of different values of ι on the satisfaction degrees of the participants: $\iota = 0$ (left) and $\iota = 1$ (right). By comparison, Fig. 6.1 (right) represented the default case where $\iota = 0.4$.

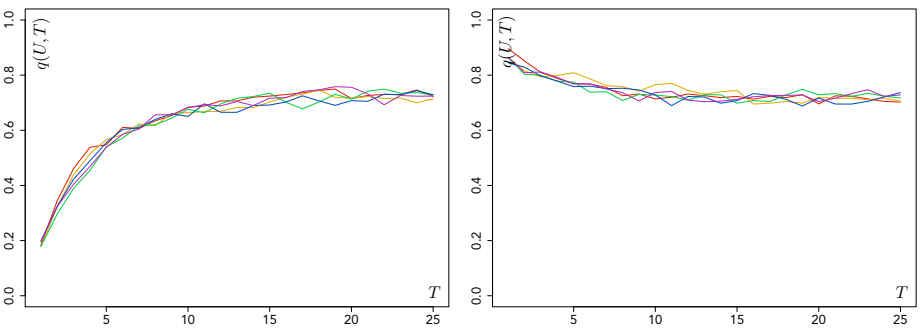

Figure 6.10: Satisfaction degrees for songs played when participants have initial satisfaction $\iota = 0$ (left) or $\iota = 1$ (right).

The graph shows that the value of ι only affects individual satisfaction for the duration of the first songs. After a while, the value is absorbed by the satisfaction decay and becomes irrelevant. After 25 songs, the satisfaction of the listeners is almost the same, independently of the value fixed for ι.

The conclusion is that the value of ι does not affect satisfaction in the long run. Using a default value of $\iota = 0.4$, slightly below the average, can help 'boost' the importance of newcomer participants, so they can immediately listen to some songs they like when entering a channel. This effect decays rapidly and, after a while, all the participants are treated equally independently of their initial satisfaction.

6.8 Summary

This chapter has presented a series of evaluations of the poolcasting technique, both with real users and artificially created scenarios.

Experiments with real users have shown that radio channels can achieve variety and smoothness but some precautions have to be taken to prevent frustrating situations, like having the same sequence of songs repeated over time, or forcing listeners to wait too long for some preferred songs. Interviewed subjects judged positively the compromise offered by the social experience in which participants do not listen only to favourite songs, but also to songs they ignore and will probably like since other like-minded listeners like them.

A series of evaluation tests has revealed how poolcasting behaves under different sets of constraints. An initial experiment was run on a set of participants with random, independent music profiles, to which poolcasting could correctly provide a customised and fair musical sequence. Although unrealistic, this experiment has shown that performance can be maintained at certain levels even in a worst-case scenario.

In a different experiment, the group was split between two discordant groups, and poolcasting could as well ensure fairness, playing songs that all the listeners liked in the long run. On the contrary, a simple average voting strategy would only satisfy the majority, leaving the minority disappointed by the music played.

In the scenario where members of the audience share musical affinity, poolcasting works even better, finding more songs the audience appreciates. The size of the audience also affects the group satisfaction, which decreases as the size decreases, although the same balance is kept among different participants even when the group is large. Satisfaction is maintained high if members of the audience tend to like the same kind of music.

Additional parameters that can affect the outcome were also evaluated (retrieval size, misery threshold, initial satisfaction) to find the most appropriate values for a good balance between performance and results.

Chapter 7

Conclusions

Not all good things come to an end
now it is only a chosen few

Elvis Costello, 1982

7.1 Summary

This dissertation has presented poolcasting, an intelligent technique that automatically programmes songs for a music channel customised for the actual audience. The goal of poolcasting is to satisfy at once a whole group of listeners, delivering a musical sequence that matches their preferences and is at the same time varied and musically smooth.

7.1.1 The goals of poolcasting

The novelty of poolcasting is the ability to autonomously perform a task commonly delegated to disc jockeys: to select in real time which songs to play according to the preferences of a group whose composition changes over time. Specifically, poolcasting compiles music programmes that fulfil four properties:

1. *Variety*: no song or artist is repeated within a short period of time;

2. *Smoothness*: songs follow a sequence perceived as musically smooth;

3. *Customisation*: songs match the musical interests of the current audience; and

4. *Fairness*: in the long run, every listener has a similar degree of satisfaction with respect to the music experienced.

7.1.2 The CBR process

The approach of poolcasting to build a group-customised musical sequence is by means of an iterated Case-Based Reasoning process that adds in real time new songs to the sequence, based on the last songs played and on the preferences of the current listeners.

Case bases

The songs available to be played at any time and the preferences of the listeners for these songs constitute the case bases. Each case is defined as a tuple ¡song, performing artist, individual preference¿.

Individual preferences are obtained from listening habits data, analysing which songs and artists each person most played and rated in the past. All the cases related to a listener form a *case base* and describe which songs an individual would (or would not) like to hear. The case bases of all the listeners at a given moment form the *collection of case bases*.

Retrieve process

When a new song has to be added to the musical sequence, poolcasting retrieves from the case bases a set of good candidate songs to be played next. Songs and artists recently played are not good candidates since they do not match the requirement of variety (Goal 1).

The remaining songs are ranked according to how well they go in sequence after the last song played. The knowledge to fulfil this task comes from the analysis of playlists collected from the Web which describe actual musical experiences of thousands of people. From their analysis, the retrieve process determines a set of songs that are more indicate to be played after the last one and that, as such, match the requirement of smoothness (Goal 2).

Reuse process

The set of retrieved candidates is then ranked according to the musical tastes of the current listeners, in order to achieve customisation (Goal 3). This process addresses the social choice problem of delivering content that is liked by the entire audience while preserving fairness in the long run.

In the reuse process, a set of individual preferences is extracted for each candidate song from the case bases. These preferences are then aggregated to determine the candidate song preferred by the group as a whole. The aggregation takes place by means of a satisfaction-weighted average: songs are ranked higher if they are most liked by less satisfied listeners. Keeping memory of past satisfactions enables the reuse process to achieve fairness (Goal 4).

Revise process

The ranked set of candidates is presented to the listeners who can adjust their preferences for each song. The expressed feedback can alter the ranking of the reuse process, making a different candidate become the one with the highest group preference.

Finally, the best ranked candidate is selected and delivered to the audience. The CBR process cycle starts again to determine the next song to add to the musical sequence.

7.1.3 The Web radio application

To evaluate the properties of poolcasting in a real scenario, the technique has been integrated into a Web radio application to provide music channels customised for their audiences.

The result is Poolcasting Web Radio, an online radio framework that offers a *social radio* experience, where the channels adapt their content for the taste of the actual audience.

The workflow of each channel follows the CBR process that characterises poolcasting. Each channel identifies a virtual space where people find a particular subset of music (e.g., a 'Rock' channel), programmed in real time according to the audience.

In this application, listeners can contribute with new songs to the repository of music by sharing their personal digital music libraries. From each shared library, the radio parses the list of songs and the listening habits data, that is, how many times each song was played and its rating. Analysing these data, the radio is able to assess music profiles for each channel listener.

When a new song has to be scheduled on a channel, the system first discards songs that do not fit the channel definition (e.g., non-Rock tracks) or that have been played recently. Next, a set of good candidate songs that go well after the last song played is retrieved and ranked according to a satisfaction-weighted average of the listeners' preferences.

Using the Web interface of the radio, the audience can browse the list of ranked candidates and revise their preferences, stating explicitly positive or negative feedback for each song. After the revision process, the best ranked candidate is finally determined as the next song to play.

7.1.4 Evaluation

A prototype of Poolcasting Web radio has been running in the local network of the IIIA and has served as the basis for a set of evaluations.

Ten persons have actively used the radio during one year; most of the opinions collected were positive with regard to the social radio experience. Listeners enjoyed the fact of listening to music with friends located elsewhere and of listening to a combination of songs they already knew with songs that were unknown to them but were preferred by like-minded listeners.

Being able to discover new music was seen as a good compromise that justified the fact of being exposed, once in a while, to music outside of their interests. Moreover, users who shared their personal libraries were able to 're-discover' songs they did not remember to have in their large collections.

A set of experiments was additionally run to evaluate the performance of poolcasting with groups made of artificial profiles with different size and musical interests. For groups that are small or whose members have a strong musical affinity, poolcasting performs particularly well, playing musical sequences that satisfy the entire audience.

The average performance tends to decrease if the group becomes larger or musically heterogeneous since poolcasting may not always find songs that rank high in *everyone*'s individual preferences. In these cases, poolcasting is still able to determine a musical sequence adapted for the audience, although with smaller average individual preferences.

Under every condition, poolcasting is able to fulfil the goal of fairness, selecting songs that, in the long run, satisfy the entire public. Even when the audience is split in two discording groups (e.g., three members love the music that two members dislike and vice versa), poolcasting is able to maintain a good balance among all the participants, playing over time songs that both sub-groups like.

These positive results are due to the satisfaction-weighted aggregation method introduced in Sect. 4.4 to combine multiple individual preferences into an iterated collective choice. This method boosts the influence of less satisfied listeners, so that their favourite songs are played within a certain amount of time even if their interests belong to a minority.

7.1.5 Possible applications

The evaluation makes clear that poolcasting can act as a good disc jockey under certain conditions, playing music that satisfies the audience and is varied and smooth. The technique can be used to customise online radio channels (as in Poolcasting Web radio), but may also be applied to other contexts.

One possible application is to automate the selection of music in a house-party. In the past, party guests would sometimes bring their own vinyl or compact discs to contribute to the music played. Nowadays, most people store their music in digital devices such as iPods. The idea of a 'poolcasting party' is to set up a party where friends bring their own iPods and connect them to a personal computer, from where music will be played. The computer runs a poolcasting system which reads the songs available on each player and autonomously generates a customised musical sequence of these songs, combining individual preferences and musical continuity. Similarly to a radio channel, the audience could restrict the music played to a specific subset (e.g., Dance music from the Nineties). As people join or leave the venue, the available music would change, and poolcasting would dynamically adapt the songs played to the current audience.

A different application for poolcasting would be to help music labels promote new releases. Music promotion on AM/FM radio is a common marketing strategy: labels pay radio stations to broadcast their latest songs, where the larger the audience of a station the higher the amount paid. The problem with this strategy is that it only works for mainstream music and is not profitable for small niches of public. With a Poolcasting Web radio, the definition of each channel would be left in the hands of the audience, according to their favourite genres, periods, tags. In this scenario, music labels would have it easier to identify very specific niches of audience that might be interested in their upcoming releases. A promotion strategy for this "long tail" [Anderson, 2004] of listeners would be cheaper and more effective than one run on a mainstream station.

7.2 Contributions

The research reported in this dissertation offers relevant contributions to different areas: Web data mining, Case-Based Reasoning, social choice and Internet radios.

7.2.1 Experiential data from the Web

The first contribution of this thesis is the demonstration of how experiential data collected from the Internet can be reused to perform a specific task.

Poolcasting is designed to generate 'good' musical sequences and this requires domain knowledge about songs and artists that go well together in sequence. Other researchers have proposed to extract this knowledge from the *content* of musical objects, analysing their acoustic features, chords or lyrics. These approaches are not quite scalable since they require either the audio content, the lyrics or a symbolic representation of each song. Moreover, these techniques can only identify songs that 'sound similar', which do not necessarily correspond to songs that go well in sequence for a particular context or group.

One goal of this research was to show how the same task is better solved observing the way in which people have *experienced* music in the past, to determine which songs may play well together in new sequences. For this purpose, about a million playlists were collected from the Web, records of the way in which people organise music for their daily activities.

The analysis of co-occurrences in these playlists has resulted in the definition of a musical association degree that allows to determine which songs and artists are more 'socially' associated.

The comparison of this association measure with other measures of similarity offered by popular music-related Web pages (Yahoo! Music, MusicStrands, All Music, Last.fm) has demonstrated that this approach is not only able to obtain equivalent results, but also to uncover relationships motivated by social or cultural reasons, rather than by acoustic ones.

Employing Web data mining to address problems that are typically solved by content-based techniques looks like a practical and effective approach, motivated by the fact that human experiences in multiple domains are becoming more and more available on the Internet.

7.2.2 Reinterpretation of CBR

The second contribution of this dissertation is the reinterpretation of the Case-Based Reasoning process and its application to a dynamic group-based scenario.

The CBR process that runs poolcasting has different innovative features. CBR has been classically understood in a single-agent framework, with *individual* past experiences stored as cases to solve new tasks. In poolcasting, past experience is obtained from *multiple* people by means of playlists and listening habits. Moreover, cases are not structured as (problem → solution) pairs, but contain the appropriate knowledge to solve the current task, that is, to determine which song to play next.

Rather than a single case base, poolcasting exploits a collection of case bases that is updated at every iteration to consider only the listeners connected at each moment. The retrieve process has been reinterpreted to determine the *most indicated* candidates to be played at a given time, according to the experiential knowledge extracted from playlists. The reuse process attacks the social choice problem of identifying the songs preferred by the group as a whole. The revise process updates the preferences of the audience and enables listeners to influence in real time the ranking of the retrieved set, determining which song will be played next.

One advantage of the CBR approach is that the four properties of variety, smoothness, customisation and fairness can be targeted in successive steps. First, poolcasting looks for candidates that address sequence-related properties appropriate for *every* type of audience (avoiding repetitions and jolting musical transitions). Then, poolcasting adapts the retrieved set to a *particular* audience. In this way, the influence of the listeners is limited to a specific range of 'good' songs that are indicate to be played in sequence.

A consequence of this approach is that, in the revise process, participants are requested to send feedback only for a small set of songs (the retrieved set), and not for the entire repository of music. This encourages people to revise their preferences; showing too many options would probably obtain the opposite effect.

Another advantage of CBR is that poolcasting can both work with 'passive' listeners, who never express feedback for the proposed songs, and with 'active' listeners. In the former case, models of musical preferences generated from the analysis of listening behaviour data are used to assess the preferences of the audience. In the latter case, the adjustments made by the listeners to their music profiles enable poolcasting to learn more precisely the interests of the audience and to improve customisation over time.

Having a collection of case bases, one for each participant, is also an advantage. Whenever people listen to music in groups, members of the audience

can join or leave at different moments. At each moment, the music should be customised only for the current members. In poolcasting, every time the CBR process is iterated, only the case bases of the current participants are considered, so that the right group-customisation is achieved. When the CBR process restarts to select the following song and the composition of the audience has changed, the collection of case bases is updated accordingly.

The characteristic of Case-Based Reasoning is to reuse past experiences to solve new tasks. Poolcasting opens a path to apply this idea to scenarios larger than a single agent, where experience is provided by a multitude of persons (as happens on the Web) and solutions are automatically adapted to the actual preferences or needs of specific users.

7.2.3 Iterated social choice

The third contribution of this thesis is the development of a strategy to iteratively aggregate multiple individual preferences in order to satisfy a group as a whole.

Poolcasting is faced with a series of consecutive decisions about which song to play at each moment. These decisions cannot be taken independently since the objective is to form a globally good sequence. The preferences of *all* the participants should be considered but, at the same time, those who have not recently listened to any favourite song should be promoted, to guarantee a balance among the entire audience.

For this purpose, a novel preference aggregation method has been introduced that determines which song to select at each moment, keeping *memory* of previous decisions.

The satisfaction-weighted preference aggregation consists of calculating the average of all the individual preferences, weighted according to the measure in which each person has enjoyed the experience so far. People who have not recently listened to any of their preferred songs obtain a larger weight and vice versa. Every time a new song is played, the weights are updated, which ensures, after a certain number of iterations, a fair and balanced satisfaction.

The rationale of this technique is that whenever people are exposed to content they do not particularly appreciate, they are soon after *rewarded* by some of their favourite music. This strategy is a trade-off between an egalitarian and an utilitarian system. The technique also includes a misery threshold which ensures that participants are never presented with songs they intensely dislike.

The satisfaction-weighted strategy is applicable to every domain where a *sequence of inter-dependent decisions* has to be taken for a group of people. Groups who meet on a regular basis to perform an activity together (watching movies, visiting places, attending restaurants, etc.) may use an adapted version of the strategy presented here for iterated social choice.

7.2.4 A social radio experience

The fourth contribution of this thesis is the development of an online application to provide group-customised music channels.

Online radios stream millions of music channels that are not affected by who is actually listening. Poolcasting Web radio takes instead advantage of the social nature of the Internet to stream music channels customised for their audience.

Customised music streams are typical of digital music services such as Last.fm and Pandora which provide *private* music channels, personalised for *individuals*. The innovation of Poolcasting Web radio is to offer *public* radio channels that anyone can join at any moment, where the content is adapted to a *group* of listeners in real time.

Poolcasting Web radio also offers innovative features to increase the social nature of the experience: listeners can vote for the proposed songs, can create new channels and, most importantly, can share their personal music libraries, contributing with music to the collection of songs broadcast on the radio. According to a recent study [McGuire and Slater, 2005], nearly one-fourth of frequent online music users say the ability to share music with others in some fashion is an important criterion when selecting an online music service.

Poolcasting Web radio combines bottom-up and top-down approaches: users can express their musical preferences while the actual choice of music played is taken by a CBR process that iteratively checks which songs are available and who is listening and selects the song most likely to satisfy the current audience.

With Poolcasting Web radio, the social component that characterises many real-world situations is integrated for the first time into a Web application. Listeners can influence the music played, chat with each other within a channel, define new channels and share personal libraries and listening habits.

Friends located around the world can meet in a virtual space created ad hoc for their musical interests and share a listening experience that is not offered by other Web services and which allows to easily get to share and discover music.

7.3 Future work

Although this dissertation has focused on customising *musical* content for a group of listeners, the techniques presented can be extended to content of different nature. The idea of 'poolcasting' comes from the words:

> pool (noun) — a combination of resources, funds, etc., for common advantage; and
>
> cast (verb) — to send forth, to deliver;

and can be defined as follows:

> **poolcast** (verb) — to collect knowledge about resources and people, and use such knowledge to deliver the right sequence of resources for the common advantage of a group of people.

Poolcasting identifies a process that does not require human intervention and works sequentially, selecting which items to deliver to the audience based on the previous items and on the preference and satisfaction of the audience.

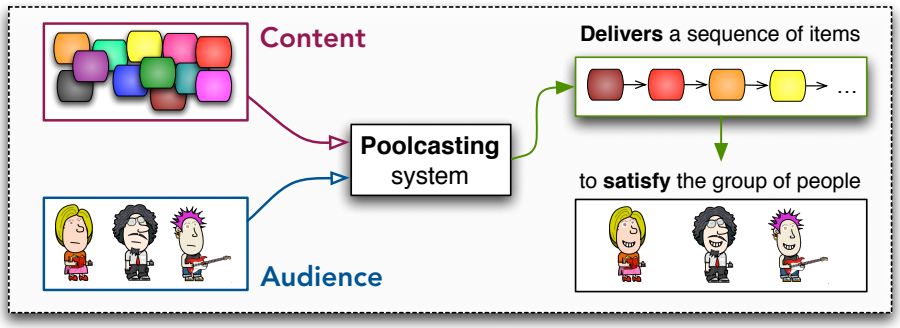

Figure 7.1: A poolcasting system scenario.

Poolcasting can be considered, similarly to collaborative filtering, as a family of techniques applicable to user-adaptive systems.

This dissertation has shown that reusing experiences collected from the Web to provide group-customised channels is a practical and productive research path. Future work will be dedicated to investigate domains other than music where this approach can result effective.

7.3.1 Generalising poolcasting

To extend poolcasting to other domains, it is important to summarise the fundamental features and requirements of the presented technique.

The first characteristic of poolcasting is its *iterative* nature. The goal of the technique is to generate a good *sequence* of items — not just one — and for this reason a selection process is iterated multiple times. At each iteration, poolcasting holds *memory* of the previous selections to ensure that the whole sequence fulfils the desired requirements.

The second property of poolcasting is its *social* nature. The technique adapts content to a group — not an individual. When members of a group have different preferences, poolcasting pursues fairness with a sequence that can possibly satisfy everyone at the same degree. Poolcasting is designed for intentional groups, who accept to collaborate rather than to compete, for whatever reason that keeps the group alive. The composition of the group can change over time and members can either be co-located or displaced.

The third characteristic is that domain knowledge comes from *reusing past experiences*. The principle is that the best way to customise content for a group is to learn from the way in which others have experienced that content in the past. In particular, the focus is on the Web which offers the largest data source for human experiences.

The last relevant property is that poolcasting does not just *recommend* but actually *delivers* adapted content (e.g., broadcasting music on the Web), enabling the audience to share an actual experience and express immediate feedback which is valuable to improve customisation over time.

Poolcasting in the domain of movies

One scenario where poolcasting could be useful is that of a family or a group of friends who meet weekly to watch movies together. Each individual might have different 'watching habits' and enjoy more or less different genres but still long for a social experience that reunites them week after week.

In this kind of situation, people implicitly accept the social compromise that they will probably not watch only movies they know and like, but will have the chance to discover titles enjoyed by their friends.

This scenario resembles that of Poolcasting Web radio, but the extension to the domain of movies requires some adjustments. While songs in a radio are played one right after the other, movies are watched in separate days, therefore their order and continuity is not as relevant as in the case of music.

The satisfaction decay also changes: listening to three consecutive 'less preferred' songs in a radio channel may be acceptable for an individual, since they would only count for 10 minutes of an entire music programme; watching three 'less preferred' movies in a row, on the other hand, would be less acceptable, since they would correspond to three weeks of negative satisfaction for a specific individual.

Another difference resides in the data collected from the Web to identify associations. Working with movies requires to gather 'watch-lists', sequences of titles watched by a person within a certain period. Similarly to music, Web communities exist that make these data available for millions of users.

The co-occurrence analysis of these data is also different. In the case of music, the simple fact of observing two songs occurring together in playlists suggests an association between them. In the case of movies, an association can be derived only when many people watched the same two movies and *rated them* identically.

A complete development of a movie-related poolcasting system remains future work, but preliminary studies suggest that the analysis of Web-mined data can produce good lists of top associated movies, as reported in Appendix A.

Poolcasting news items

Another promising scenario for poolcasting is delivering news items to a group of displaced friends. Nowadays there is a clear discrepancy in the way in which news are produced and consumed. Newspapers, newscasts and magazines force users to experience news in a fixed order, determined by the editorial board to satisfy an imaginary 'average' consumer (e.g., first politics, then economics, then weather, then sport). However every person has a particular way of experiencing news; for instance one might flip through the first pages of a newspapers and skip directly to sport and weather.

Applying the poolcasting strategy to news items would mean to let people organise themselves in groups (*news channels*) where they would specify their interests either explicitly (by means of genres, periods, tags) or implicitly (inferred from their previous experiences).

Within a channel, people would share news items with others and, in return,

discover news that interest other like-minded persons. News producers would benefit from this model since they would be able to deliver the right *content* to the right *target*.

Poolcasting news items would also blunt geographical constraints. People would be able to access their own 'local news' channels, independently from their actual location. An Indian citizen living abroad, for instance, would still be able to receive daily updates about his hometown and become aware of news items prompted by his friends in India.

Poolcasting TV programmes

A different domain where to apply poolcasting is that of a group of people, sitting together in front of a TV set, who have to decide which programmes to watch. Thanks to PVRs (Personal Video Recorders), viewers can nowadays *personalise* the order in which they wish to experience TV shows, but these systems only work for one person at the time and do not allow a *group* to find a sequence of programmes that can satisfy everyone.

A common situation is that of a family where the father likes sport and action/comedy movies, the mother likes comedy movies and quiz shows, the grandmother likes documentaries and talk shows, the kid likes animated series and comedies: what programmes should they watch together? Is there a social alternative to splitting the family into four separate screens?

The idea of a 'poolcasting TV' consists in a system that autonomously configures prime-time TV group sessions day after day. This system would first build user profiles of each spectator, based on explicit feedback and observed behaviours (e.g., which shows each person tends to watch, at which time, for how long, when is the person at home). During each session, the system would then aggregate multiple preferences, trying to satisfy the whole family in the long run. For instance, one night the system might first broadcast an episode of an animated series, so the kid can happily go to sleep; then a documentary about sport legends, to generate an enriching discussion between father and grandmother; then a comedy movie, which would fairly satisfy all the adults. Memory of past choices would help achieve balance and fairness among all the family members along time.

7.3.2 Improving poolcasting

A different direction for future work deals with ways to improve the experience of members of poolcasting channels.

Self-adaptive channels

A possible improvement to poolcasting is to introduce self-adaptive channels. These channels would diagnose decreases in performance and react accordingly, suggesting a solution to the participants.

An example is given by the situation where a Poolcasting Web radio channel is listened by two discordant groups of listeners and the system has to struggle to adapt the music for the entire audience. In this case, a self-adaptive channel would detect the situation and suggest the two sub-groups to split into two separate channels, in order to maintain a higher degree of group satisfaction.

Introducing self-adaptive channels would be of great advantage to the participants. Self-adaptive channels can be seen as form of *goal-driven learning* [Leake and Ram, 1993] as they would recognise situations where some type of failure occurs, access a library of possible correcting actions and then apply the best suited one to fix the problem.

Fuzzy affinity degrees

A different direction to improve poolcasting experience is to enrich *music ontology* from simple Boolean categories to fuzzy affinity degrees.

Poolcasting can currently categorise songs and artists in the music pool only in Boolean terms. Madonna, for instance, *belongs to* the genre Pop and *not to* the genres R&B or Jazz. Likewise, Madonna *has been tagged as* 'female pop' and *not as* 'Spanish pop'.

The limitation of this representation becomes clear when poolcasting has to schedule music for multi-genre channels, for instance for a radio channel defined as 'genre in (Pop, R&B)'. Ideally, a good musical selection for this channel would be to first play a song from the *core* of a genre (e.g., a Pop song), then a series of songs *in between* two genres (e.g., Pop/R&B tracks) to finally reach a song in the core of the second genre and gradually move back to the first genre. In order to achieve this behaviour, the music ontology employed by poolcasting has to be enriched to include *fuzzy membership degrees*, to measure the *degree* in which each song or artist belongs to each genre or tag.

A method to describe songs and artists in terms of fuzzy membership degrees was introduced by the author in [Baccigalupo et al., 2008]. Thanks to this method, artists can be characterised as vectors of fuzzy membership values rather than simply as Boolean categories. The music of Madonna, for instance, can be described as belonging to different genres at different fuzzy degrees: Pop (0.4), R&B (0.8), Jazz (0.3), and so on.

Future work will be dedicated to embody the concept of "genre affinity degree" into poolcasting. Users would benefit from this approach as they would be able to specify whether a channel should only play *core* artists of a given genre (who show a high membership degree) or whether *cross-genre* artists are allowed as well.

Poolcasting could also be able to better programme multi-genre radio channels, playing music that smoothly shifts from the core of a genre, to cross-genre songs, to songs belonging to a different genre and so on. This improvement would make poolcasting behave more similarly to professional disc jockeys, who are able to swiftly transport music from a style/genre/tempo to a different one without disrupting transitions.

7.4 Related publications

The following papers were published as part of this research.

[Baccigalupo and Plaza, 2006] C. Baccigalupo and E. Plaza. Case-based sequential ordering of songs for playlist recommendation. In T. Roth-Berghofer, M.H. Göker, and H.A. Güvenir, editors, *Advances in Case-Based Reasoning, Proceedings of the 8th European Conference on Case-Based Reasoning (ECCBR 2006)*, volume 4106 of *Lecture Notes in Computer Science*, pages 286–300. Springer, 2006.

[Baccigalupo and Plaza, 2007a] C. Baccigalupo and E. Plaza. A case-based song scheduler for group customised radio. In R. Weber and M.M. Richter, editors, *Case-Based Reasoning Research and Development, Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR 2007)*, volume 4626 of *Lecture Notes in Computer Science*, pages 433–448. Springer, 2007.

[Baccigalupo and Plaza, 2007b] C. Baccigalupo and E. Plaza. Mining music social networks for automating social music services. In *Workshop Notes of the ECML/PKDD 2007 Workshop on Web Mining 2.0*, pages 123–134, 2007.

[Baccigalupo and Plaza, 2007c] C. Baccigalupo and E. Plaza. Poolcasting: A social Web radio architecture for group customisation. In *Proceedings of the Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS '07)*, pages 115–122. IEEE Computer Society, 2007.

[Baccigalupo and Plaza, 2007d] C. Baccigalupo and E. Plaza. Sharing and combining listening experience: a social approach to Web radio. In *Proceedings of the 2007 International Computer Music Conference (ICMC)*, pages 228–231, 2007.

[Baccigalupo et al., 2008] C. Baccigalupo, E. Plaza, and J. Donaldson. Uncovering affinity of artists to multiple genres from social behaviour data. In ISMIR [ISMIR, 2008], pages 275–280.

[Plaza and Baccigalupo, 2009] E. Plaza and C. Baccigalupo. Principle and praxis in the experience Web: a case study in social music. In S.J. Delany, editor, *Proceedings of the ICCBR 2009 Workshops*, pages 55–63. University of Washington Tacoma, 2009.

Bibliography

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59.

[Ahonen-Myka, 1999] Ahonen-Myka, H. (1999). Finding all frequent maximal sequences in text. *Proceedings of the 16th International Conference on Machine Learning ICML-99 Workshop on Machine Learning in Text Data Analysis*, pages 11–17.

[Anderson, 2004] Anderson, C. (2004). The long tail. *Wired*, 12(10):170–177.

[Andric and Haus, 2006] Andric, A. and Haus, G. (2006). Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications*, 29(2):127–151.

[Ardissono et al., 2003] Ardissono, L., Gena, C., Torasso, P., Bellifemine, F., Chiarotto, A., Difino, A., and Negro, B. (2003). Personalized recommendation of TV programs. *Lecture Notes in Computer Science*, pages 474–486.

[Arrow, 1970] Arrow, K. (1970). *Social Choice and Individual Values*. Yale University Press.

[Aucouturier and Pachet, 2002] Aucouturier, J. and Pachet, F. (2002). Music similarity measures: what's the use? In [ISMIR, 2002].

[Aucouturier and Pachet, 2004] Aucouturier, J. and Pachet, F. (2004). Improving timbre similarity: How high is the sky. *Journal of Negative Results in Speech and Audio Sciences*, 1(1):1–13.

[Baccigalupo and Plaza, 2006] Baccigalupo, C. and Plaza, E. (2006). Case-based sequential ordering of songs for playlist recommendation. In Roth-Berghofer, T., Göker, M., and Güvenir, H., editors, *Advances in Case-Based Reasoning, Proceedings of the 8th European Conference on Case-Based Reasoning (ECCBR 2006)*, volume 4106 of *Lecture Notes in Computer Science*, pages 286–300. Springer.

[Baccigalupo and Plaza, 2007a] Baccigalupo, C. and Plaza, E. (2007a). A case-based song scheduler for group customised radio. In Weber, R. and Richter,

M., editors, *Case-Based Reasoning Research and Development, Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR 2007)*, volume 4626 of *Lecture Notes in Computer Science*, pages 433–448. Springer.

[Baccigalupo and Plaza, 2007b] Baccigalupo, C. and Plaza, E. (2007b). Mining music social networks for automating social music services. In *Workshop Notes of the ECML/PKDD 2007 Workshop on Web Mining 2.0*, pages 123–134.

[Baccigalupo and Plaza, 2007c] Baccigalupo, C. and Plaza, E. (2007c). Poolcasting: a social Web radio architecture for group customisation. In *Proceedings of the Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS '07)*, pages 115–122. IEEE Computer Society.

[Baccigalupo and Plaza, 2007d] Baccigalupo, C. and Plaza, E. (2007d). Sharing and combining listening experience: a social approach to Web radio. In *Proceedings of the 2007 International Computer Music Conference (ICMC)*, pages 228–231.

[Baccigalupo et al., 2008] Baccigalupo, C., Plaza, E., and Donaldson, J. (2008). Uncovering affinity of artists to multiple genres from social behaviour data. In [ISMIR, 2008], pages 275–280.

[Baelde and Mimram, 2008] Baelde, D. and Mimram, S. (2008). De la webradio lambda à la λ-webradio. *Journées Francophones des Langages Applicatifs*, pages 47–62.

[Banerjee, 1992] Banerjee, A. (1992). A simple model of herd behavior. *Quarterly Journal of Economics*, 107(3):797–817.

[Baumann and Halloran, 2004] Baumann, S. and Halloran, J. (2004). An ecological approach to multimodal subjective music similarity perception. In *Proceedings of the Conference in Interdisciplinary Musicology (CIM)*.

[Bekkerman et al., 2006] Bekkerman, P., Kraus, S., and Ricci, F. (2006). Applying cooperative negotiation methodology to group recommendation problem. In *Proceeding of the ECAI-2006 Workshop on Recommender systems*, pages 72–75.

[Bennett and Lanning, 2007] Bennett, J. and Lanning, S. (2007). The Netflix prize. In *Proceedings of KDD Cup and Workshop*.

[Berenzweig and Ellis, 2001] Berenzweig, A. and Ellis, D. (2001). Locating singing voice segments within music signals. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, volume 2001, pages 1–4.

[Berenzweig et al., 2002] Berenzweig, A., Ellis, D., and Lawrence, S. (2002). Using voice segments to improve artist classification of music. In *AES 22nd International Conference*.

[Berners-Lee, 1991] Berners-Lee, T. (1991). WorldWideWeb - Executive summary. Newsletter discussion. http://groups.google.com/group/alt.hypertext/msg/395f282a67a1916c.

[Billsus and Pazzani, 2007] Billsus, D. and Pazzani, M. (2007). Adaptive news access. In [Brusilovsky et al., 2007], pages 550–570.

[Booth et al., 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web services architecture. *W3C Working Group Note*, 11:2005–1.

[Brown et al., 2001] Brown, B., Sellen, A., and Geelhoed, E. (2001). Music sharing as a computer supported collaborative application. In *Proceedings of the 7th European Conference on Computer Supported Cooperative Work*, pages 179–198. Kluwer Academic Publishers.

[Brusilovsky, 2001] Brusilovsky, P. (2001). Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110.

[Brusilovsky et al., 2007] Brusilovsky, P., Kobsa, A., and Nejdl, W., editors (2007). *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*. Springer.

[Burke, 1999] Burke, R. (1999). The Wasabi Personal Shopper: A case-based recommender system. In *Proceedings of the 11th National Conference on Innovative Applications of Artificial Intelligence*, pages 844–849.

[Burke, 2000] Burke, R. (2000). Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems*, 69(Supplement 32):175–186.

[Burke et al., 1997] Burke, R., Hammond, K., and Young, B. (1997). The FindMe approach to assisted browsing. *IEEE Expert*, 12(4):32–40.

[Cano et al., 2005] Cano, P., Batlle, E., Kalker, T., and Haitsma, J. (2005). A review of audio fingerprinting. *The Journal of VLSI Signal Processing*, 41(3):271–284.

[CBS Corporation, 2007] CBS Corporation (2007). CBS corporation acquires Last.fm, a community-based, music discovery network with a global reach. Press Release. http://www.cbscorporation.com/news/prdetails.php?id=2263.

[Chae and Flores, 1998] Chae, S. and Flores, D. (1998). Broadcasting versus narrowcasting. *Information Economics and Policy*, 10(1):41–57.

[Chaffee and Metzger, 2001] Chaffee, S. and Metzger, M. (2001). The end of mass communication? *Mass Communication & Society*, 4(4):365–379.

[Chen and Chen, 2001] Chen, H. and Chen, A. (2001). A music recommendation system based on music data grouping and user interests. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 231–238. ACM Press New York, NY, USA.

[Chen et al., 2008] Chen, Y., Cheng, L., and Chuang, C. (2008). A group recommendation system with consideration of interactions among group members. *Expert systems with applications*, 34(3):2082–2090.

[Chevaleyre et al., 2007] Chevaleyre, Y., Endriss, U., Lang, J., and Maudet, N. (2007). A short introduction to computational social choice. *Lecture Notes in Computer Science*, 4362:51.

[Chun and Hong, 2001] Chun, I. and Hong, I. (2001). The implementation of knowledge-based recommender system forelectronic commerce using Java expert system library. In *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE)*, volume 3.

[Claypool et al., 2001] Claypool, M., Brown, D., Le, P., and Waseda, M. (2001). Inferring user interest. *Internet Computing, IEEE*, 5(6):32–39.

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

[comScore, Inc., 2009] comScore, Inc. (2009). Global internet audience surpasses 1 billion visitors. World Wide Web electronic publication. `http://www.comscore.com/Press_Events/Press_Releases/2009/1/Global_Internet_Audience_1_Billion/(language)/eng-US`.

[Cosley et al., 2003] Cosley, D., Lam, S., Albert, I., Konstan, J., and Riedl, J. (2003). Is seeing believing? How recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI conference on Human factors in Computing Systems*, pages 585–592. ACM New York, NY, USA.

[Creative Commons, 2004] Creative Commons (2004). Licenses. World Wide Web electronic publication. `http://creativecommons.org/about/licenses/`.

[Crossen et al., 2002] Crossen, A., Budzik, J., and Hammond, K. (2002). Flytrap: intelligent group music recommendation. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 184–185. ACM Press New York, NY, USA.

[Cunningham et al., 2001] Cunningham, P., Bergmann, R., Schmitt, S., Traphöner, R., Breen, S., and Smyth, B. (2001). Websell: Intelligent sales assistants for the World Wide Web. *Künstliche Intelligenz (KI)*, 15(1):28–32.

[Cunningham et al., 2004] Cunningham, S., Bainbridge, D., and Falconer, A. (2004). 'More of an art than a science': supporting the creation of playlists and mixes. In [ISMIR, 2004].

[Díaz et al., 2006] Díaz, F., Fdez-Riverola, F., and Corchado, J. (2006). gene-CBR: a Case-Based Reasoning tool for cancer diagnosis using microarray data sets. *Computational Intelligence*, 22(3-4):254–268.

[Dieterich et al., 1993] Dieterich, H., Malinowski, U., Kuehme, T., and Schneider-Hufschmidt, M. (1993). State of the art in adaptive user interfaces. *Human factors in Information Technology*, 10:13–13.

[Ellis, 2003] Ellis, D. (2003). Art of the mix playlist data statistics. World Wide Web electronic publication. `http://labrosa.ee.columbia.edu/projects/musicsim/aotm.html`.

[Fox, 2007] Fox, A. (2007). Battle of the music recommender systems: User-centered evaluation of collaborative filtering, content-based analysis and hybrid systems. Master's thesis, School of Information and Library Science.

[Gates et al., 2006] Gates, C., Subramanian, S., and Gutwin, C. (2006). DJs' perspectives on interaction and awareness in nightclubs. In *Proceedings of the 6th conference on Designing Interactive Systems*, pages 70–79. ACM New York, NY, USA.

[Giles, 2005] Giles, J. (2005). Internet encyclopaedias go head to head. *Nature*, 438:900–901.

[Goldberg et al., 1992] Goldberg, D., Nichols, D., Oki, B., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

[Goldberg et al., 2001] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151.

[Gonze et al., 2006] Gonze, L., Friedrich, M., and Kaye, R. (2006). XSPF version 1. World Wide Web electronic publication. `http://xspf.org/xspf-v1.html`.

[Grabisch et al., 1998] Grabisch, M., Orlovski, S., and Yager, R. (1998). Fuzzy aggregation of numerical preferences. *Fuzzy Sets in Decision Analysis, Operations Research and Statistics*.

[Haseman et al., 2002] Haseman, W., Nuipolatoglu, V., and Ramamurthy, K. (2002). An empirical investigation of the influences of the degree of interactivity on user-outcomes in a multimedia environment. *Information Resources Management Journal*, 15(2):31–48.

[Hauver and French, 2001] Hauver, D. and French, J. (2001). Flycasting: Using collaborative filtering to generate a playlist for online radio. In *Proceedings of the International Conference on Web Delivery of Music (WEDELMUSIC)*, pages 123–130, Florence, Italy. IEEE Computer Society Press.

[Hayes and Cunningham, 2001] Hayes, C. and Cunningham, P. (2001). Smart Radio — Community based music radio. *Knowledge-Based Systems*, 14(3-4):197–201.

[Herlocker et al., 2004] Herlocker, J., Konstan, J., Terveen, L., and Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53.

[Hill et al., 1995] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

[Hirst et al., 1997] Hirst, G., DiMarco, C., Hovy, E., and Parsons, K. (1997). Authoring and generating health-education documents that are tailored to the needs of the individual patient. *Courses and lectures — International Centre from Mechanical Sciences*, pages 107–118.

[Hofmann, 2004] Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115.

[Hofmann and Puzicha, 1998] Hofmann, T. and Puzicha, J. (1998). Statistical models for co-occurrence data.

[Hogg, 1996] Hogg, M. (1996). Social identity, self-categorization, and the small group. *Understanding group behavior*, 2:227–253.

[Hohl et al., 1996] Hohl, H., Böcker, H., and Gunzenhäuser, R. (1996). Hypadapter: An adaptive hypertext system for exploratory learning and programming. *User Modeling and User-Adapted Interaction*, 6(2):131–156.

[Holland, 1975] Holland, J. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. The MIT Press.

[Horvitz et al., 1998] Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 256–265.

[Hu and Downie, 2007] Hu, X. and Downie, J. (2007). Exploring mood metadata: Relationships with genre, artist and usage metadata. In [ISMIR, 2007], pages 67–72.

[ISMIR, 2002] ISMIR (2002). *Proceedings of the 3rd International Conference on Music Information Retrieval*, Paris, France.

[ISMIR, 2004] ISMIR (2004). *Proceedings of the 5th International Conference on Music Information Retrieval*, Barcelona, Spain.

[ISMIR, 2007] ISMIR (2007). *Proceedings of the 8th International Conference on Music Information Retrieval*, Vienna, Austria.

[ISMIR, 2008] ISMIR (2008). *Proceedings of the 9th International Conference on Music Information Retrieval*, Philadelphia, Pennsylvania USA.

[ISRC Agency, 2003] ISRC Agency, I. (2003). *International Standard Recording Code (ISRC) Handbook*. IFPI Secretariat, 2nd edition.

[Jameson, 2004] Jameson, A. (2004). More than the sum of its members: challenges for group recommender systems. In *Proceedings of the Working Conference on Advanced Visual interfaces*, pages 48–54. ACM Press New York, NY, USA.

[Jameson and Smyth, 2007] Jameson, A. and Smyth, B. (2007). Recommendation to groups. *Lecture Notes in Computer Science*, 4321:596.

[Johnson and Kaye, 2004] Johnson, T. and Kaye, B. (2004). Wag the blog: how reliance on traditional media and the internet influence credibility perceptions of Weblogs among blog users. *Journalism and Mass Communication Quarterly*, 81(3):622–642.

[Kaji et al., 2005] Kaji, K., Hirata, K., and Nagao, K. (2005). A music recommendation system based on annotations about listeners' preferences and situations. In *Proceedings of the First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'05)*, pages 231–234. IEEE Computer Society.

[Kass and Finin, 1988] Kass, R. and Finin, T. (1988). Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22.

[Kelly and Teevan, 2003] Kelly, D. and Teevan, J. (2003). Implicit feedback for inferring user preference: a bibliography. In *ACM SIGIR Forum*, volume 37, pages 18–28. ACM New York, NY, USA.

[Kleedorfer et al., 2008] Kleedorfer, F., Knees, P., and Pohle, T. (2008). Oh Oh Oh Whoah! Towards automatic topic detection in song lyrics. In [ISMIR, 2008], pages 287–292.

[Kobsa, 2001] Kobsa, A. (2001). Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11(1):49–63.

[Kolodner, 1993] Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann Publishers Inc.

[Krulwich and Burkey, 1996] Krulwich, B. and Burkey, C. (1996). Learning user information interests through extraction of semantically significant phrases. In *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*, pages 100–112.

[Kuo and Shan, 2002] Kuo, F. and Shan, M. (2002). A personalized music filtering system based on melody style classification. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 649–652.

[Lang, 1995] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*.

[Leake and Ram, 1993] Leake, D. and Ram, A. (1993). Goal-driven learning: fundamental issues. *AI Magazine*, 14(4):67–72.

[Leitich and Toth, 2007] Leitich, S. and Toth, M. (2007). PublicDJ — Music selection in public spaces as multiplayer game. In *Proceedings of Audio Mostly 2007*.

[Levy and Sandler, 2007] Levy, M. and Sandler, M. (2007). A semantic space for music derived from social tags. In [ISMIR, 2007], pages 411–416.

[Lieberman, 1995] Lieberman, H. (1995). Letizia: An agent that assists Web browsing. In *Internation Joint Conference on Artificial Intelligence*, volume 14, pages 924–929. Lawrence Erlbaum Associates Ltd.

[List and Pettit, 2002] List, C. and Pettit, P. (2002). Aggregating sets of judgments: an impossibility result. *Economics and Philosophy*, 18(01):89–110.

[Liu and Maes, 2005] Liu, H. and Maes, P. (2005). InterestMap: harvesting social network profiles for recommendations. In *Proceedings of the IUI 2005 Beyond Personalization Workshop*, pages 54–59.

[Logan et al., 2003] Logan, B., Ellis, D., and Berenzweig, A. (2003). Toward evaluation techniques for music similarity. *The MIR/MDL Evaluation Project White Paper Collection*, 3:81–85.

[López De Mántaras et al., 2005] López De Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M., Aamodt, A., and Watson, I. (2005). Retrieval, reuse, revision and retention in Case-Based Reasoning. *The Knowledge Engineering Review*, 20(3):215–240.

[Magno and Sable, 2008] Magno, T. and Sable, C. (2008). A comparison of signal-based music recommendation to genre labels, collaborative filtering, musicological analysis, human recommendation, and random baseline. In [ISMIR, 2008], pages 161–166.

[Mahedero et al., 2005] Mahedero, J., Martínez, Á., Cano, P., Koppenberger, M., and Gouyon, F. (2005). Natural language processing of lyrics. In *Proceedings of the 13th annual ACM International Conference on Multimedia*, pages 475–478. ACM New York, NY, USA.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289.

[Marling and Whitehouse, 2001] Marling, C. and Whitehouse, P. (2001). Case-Based Reasoning in the care of Alzheimer's disease patients. *Lecture Notes in Computer Science*, pages 702–715.

[Martin, 2008] Martin, S. (2008). DrL: Distributed Recursive (Graph) Layout. World Wide Web electronic publication. `http://www.cs.sandia.gov/~smartin/software.html`.

[Masthoff, 2004] Masthoff, J. (2004). Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction*, 14(1):37–85.

[Masthoff and Gatt, 2006] Masthoff, J. and Gatt, A. (2006). In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. *User Modeling and User-Adapted Interaction*, 16(3):281–319.

[Mayer et al., 2008] Mayer, R., Neumayer, R., and Rauber, A. (2008). Rhyme and style features for musical genre classification by song lyrics. In [ISMIR, 2008], pages 337–342.

[McCarthy, 2002] McCarthy, J. (2002). Pocket RestaurantFinder: A situated recommender system for groups. In *Workshop on Mobile Ad-Hoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems*.

[McCarthy and Anagnost, 1998] McCarthy, J. and Anagnost, T. (1998). MusicFX: an arbiter of group preferences for computer supported collaborative workouts. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, pages 363–372. ACM Press New York, NY, USA.

[McCarthy et al., 2006] McCarthy, K., McGinty, L., Smyth, B., and Salamó, M. (2006). Social interaction in the CATS group recommender. In Brusilovsky, P., Dron, J., and Kurhila, J., editors, *Workshop on the Social Navigation and Community-Based Adaptation Technologies at the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*.

[McEnnis and Cunningham, 2007] McEnnis, D. and Cunningham, S. (2007). Sociology and music recommendation systems. In [ISMIR, 2007], pages 185–190.

[McGuire and Slater, 2005] McGuire, M. and Slater, D. (2005). Consumer taste sharing is driving the online music business and democratizing culture. *The Berkman Center for Internet & Society, Harvard Law School*.

[Messick and Brewer, 1983] Messick, D. M. and Brewer, M. B. (1983). Solving social dilemmas: a review. In Wheeler, L. and Shaver, P., editors, *Review of personality and social psychology*, volume 4, pages 11–44. Sage.

[MIDI Manufacturers Association, 1996] MIDI Manufacturers Association, T. (1996). *The Complete MIDI 1.0 Detailed Specification*. The MIDI Manufacturers Association.

[Miller, 2009] Miller, F. (2009). Message from the Last.fm founders, Felix, RJ and Martin. World Wide Web electronic publication. `http://blog.last.fm/2009/06/10/message-from-the-lastfm-founders-felix-rj-and-martin`.

[Montaner et al., 2003] Montaner, M., López, B., and de la Rosa, J. (2003). A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19(4):285–330.

[Mufin, 2008] Mufin (2008). Mufin - about us. World Wide Web electronic publication. `http://www.mufin.com/gb/about.html`.

[Nakagawa and Mobasher, 2003] Nakagawa, M. and Mobasher, B. (2003). Impact of site characteristics on recommendation models based on association rules and sequential patterns. In *Proceedings of the IJCAI*, volume 3.

[News Corporation, 2005] News Corporation (2005). News Corporation to acquire Intermix Media, Inc. Press Release. `http://www.newscorp.com/news/news_251.html`.

[Nichols, 1997] Nichols, D. (1997). Implicit rating and filtering. In *Proceedings of 5th DELOS Workshop on Filtering and Collaborative Filtering*, pages 31–36.

[Novemsky and Dhar, 2005] Novemsky, N. and Dhar, R. (2005). Goal fulfillment and goal targets in sequential choice. *Journal of Consumer Research*, 32(3):396–404.

[NPD Group, 2005] NPD Group, T. (2005). Computer listening behaviors on the rise. World Wide Web electronic publication. `http://www.npd.com/press/releases/press_050512.html`.

[Oard and Kim, 2001] Oard, D. and Kim, J. (2001). Modeling information content using observable behavior. In *Proceedings of the annual meeting – American Society for Information Science*, volume 38, pages 481–488. Information Today.

[O'Connor et al., 2001] O'Connor, M., Cosley, D., Konstan, J., and Riedl, J. (2001). PolyLens: a recommender system for groups of users. In *Proceedings of the 7th European Conference on Computer Supported Cooperative Work*, pages 199–218, Norwell, MA, USA. Kluwer Academic Publishers.

[O'Hara et al., 2004] O'Hara, K., Lipson, M., Jansen, M., Unger, A., Jeffries, H., and Macer, P. (2004). Jukola: democratic music choice in a public space. In *Proceedings of the 2004 conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 145–154, New York, NY, USA. ACM Press.

[Ouko et al., 2002] Ouko, L., Sipitakiat, A., Gomez-Monroy, C., and DiMicco, J. (2002). The Common Sense Disc Jockey. Technical report, MIT Media Lab.

[Pachet and Cazaly, 2000] Pachet, F. and Cazaly, D. (2000). A taxonomy of musical genres. In *Proc. Content-Based Multimedia Information Access (RIAO)*, pages 1238–1245.

[Pachet et al., 2001] Pachet, F., Westermann, G., and Laigre, D. (2001). Musical data mining for electronic music distribution. In *Proceedings of the 1st International Conference on Web Delivering of Music (WEDELMUSIC)*, pages 101–106.

[Pandora Media, Inc., 2006] Pandora Media, Inc. (2006). About the music genome project®. World Wide Web electronic publication. `http://www.pandora.com/corporate/mgp`.

[Plaza and Baccigalupo, 2009] Plaza, E. and Baccigalupo, C. (2009). Principle and praxis in the experience Web: a case study in social music. In Delany, S., editor, *Proceedings of the ICCBR 2009 Workshops*, pages 55–63. University of Washington Tacoma.

[Potter, 2008] Potter, G. (2008). Putting the collaborator back into collaborative filtering. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[Raghunathan and Corfman, 2008] Raghunathan, R. and Corfman, K. (2008). Is happiness shared doubled and sadness shared halved? social influence on enjoyment of hedonic experiences. *Journal of Marketing Research*.

[Ragno et al., 2005] Ragno, R., Burges, C., and Herley, C. (2005). Inferring similarity between music objects with application to playlist generation. In *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 73–80, New York, NY, USA. ACM Press.

[Raimond and Sandler, 2007] Raimond, Y. and Sandler, M. (2007). Using the semantic Web for enhanced audio experiences. In *Proceedings of the 123rd Audio Engineering Society (AES) Convention*.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of net news. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 175–186.

[Rich, 1979] Rich, E. (1979). *Building and Exploiting User Models*. PhD thesis, Carnegie Mellon University.

[Rose and Lenski, 2008] Rose, B. and Lenski, J. (2008). The Infinite Dial 2008: Radio's digital platforms. World Wide Web electronic publication. `http://www.arbitron.com/downloads/digital_radio_study_2008.pdf`.

[Saklofske and Yackulic, 1989] Saklofske, D. and Yackulic, R. (1989). Personality predictors of loneliness. *Personality and individual differences*, 10(4):467–472.

[Schedl et al., 2006] Schedl, M., Pohle, T., Knees, P., and Widmer, G. (2006). Assigning and visualizing music genres by web-based co-occurrence analysis. Victoria, Canada.

[Shardanand, 1994] Shardanand, U. (1994). *Social Information Filtering for Music Recommendation*. PhD thesis, Massachusetts Institute of Technology.

[Shimazu, 2001] Shimazu, H. (2001). ExpertClerk: Navigating shoppers' buying process with the combination of asking and proposing. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 1443–1450. Lawrence Erlbaum Associates Ltd.

[Shur and Hangartner, 2006] Shur, J. and Hangartner, R. (2006). Playlist evaluation. World Wide Web electronic publication. `http://blog.recommenders06.com/wp-content/uploads/2006/09/shur2.pdf`.

[Smyth, 2007] Smyth, B. (2007). Case-Based recommendation. In [Brusilovsky et al., 2007], pages 342–376.

[Sociedad General de Autores y Editores, 2006] Sociedad General de Autores y Editores (2006). Tarifas generales SGAE. World Wide Web electronic publication. `http://www.sgae.es/tipology/est/item/es/1025_291.html`.

[Sordo et al., 2008] Sordo, M., Celma, O., Blech, M., and Guaus, E. (2008). The quest for musical genres: do the experts and the wisdom of crowds agree? In *Proceedings of the 9th International Conference on Music Information Retrieval*, page 255.

[Sprague et al., 2008] Sprague, D., Wu, F., and Tory, M. (2008). Music selection using the PartyVote democratic jukebox. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 433–436, New York, NY, USA. ACM Press.

[Strands, Inc., 2006] Strands, Inc. (2006). So, what is partyStrands? World Wide Web electronic publication. http://blog.strands.com/2006/10/01/so-what-is-partystrands/.

[Sunstein, 2005] Sunstein, C. (2005). Group judgments: Deliberation, statistical means, and information markets. *New York University Law Review*, 80:962–1049.

[Surowiecki, 2004] Surowiecki, J. (2004). *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Doubleday.

[Tangerine, 2009] Tangerine (2009). Tangerine! — Potion factory. World Wide Web electronic publication. http://www.potionfactory.com/tangerine/.

[Terveen and McDonald, 2005] Terveen, L. and McDonald, D. (2005). Social matching: a framework and research agenda. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(3):401–434.

[The Echo Nest, 2009] The Echo Nest (2009). The Echo Nest — Recommend. World Wide Web electronic publication. http://the.echonest.com/recommend/.

[Tindale et al., 2003] Tindale, R., Kameda, T., and Hinsz, V. (2003). Group decision making. *Sage handbook of social psychology*, pages 381–403.

[Togelius et al., 2006] Togelius, J., De Nardi, R., and Lucas, S. (2006). Making racing fun through player modeling and track evolution. In *Workshop on Adaptive Approaches for Optimizing Player Satisfaction in Computer and Physical Games (SAB)*, pages 61–70.

[Tzanetakis and Cook, 2002] Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302.

[van Dijck, 2006] van Dijck, J. (2006). Record and hold: popular music between personal and collective memory. *Critical Studies in Media Communication*, 23(5):357–374.

[Vignoli, 2004] Vignoli, F. (2004). Digital music interaction concepts: a user study. In [ISMIR, 2004].

[Voida et al., 2006] Voida, A., Grinter, R., and Ducheneaut, N. (2006). Social practices around iTunes. In O'Hara, K. and Brown, B., editors, *Consuming music together: social and collaborative aspects of music consumption technologies*, pages 57–83. Springer.

[Wang, 1997] Wang, K. (1997). Discovering patterns from large and dynamic sequential data. *Journal of Intelligent Information Systems*, 9(1):33–56.

[Weber-Lee et al., 1997] Weber-Lee, R., Barcia, R., Da Costa, M., Rodrigues Filho, I., Hoeschl, H., D'Agostini Bueno, T., Martins, A., and Pacheco, R. (1997). A large Case-Based reasoner for legal cases. *Lecture Notes in Computer Science*, pages 190–199.

[Weiss, 2000] Weiss, A. (2000). Music selection for Internet radio. Technical Report COSC460, Department of Computer Science, University of Canterbury, New Zealand.

[Whitman and Lawrence, 2002] Whitman, B. and Lawrence, S. (2002). Inferring descriptions and similarity for music from community metadata. In [ISMIR, 2002], pages 591–598.

[WXYC, 2004] WXYC (2004). WXYC Simulcast. World Wide Web electronic publication. `http://wxyc.org/about/first/`.

[Zadel and Fujinaga, 2004] Zadel, M. and Fujinaga, I. (2004). Web services for music information retrieval. In [ISMIR, 2004].

[Zhang et al., 2002] Zhang, J., Mostafa, J., and Bloomington, I. (2002). Comparing two approaches of generating interest profiles for information filtering: Interest inferred from typical user actions versus rating of content. In *Proceedings of the 30th Annual Conference of the Canadian Association for Information Science*.

Appendix A

Top-associated movies

Preliminary experiments have shown that the analysis of watch-lists from the Web can return valuable lists of top associated movies, applying the same co-occurrence analysis process reported in Sect. 2.3 to uncover associated songs. The idea is that the more users have *rented* the same two movies within a short period and have *liked* them both, the more those movies are associated and can be recommended to people who have only watched one of the two.

Experiments were run on a set of watch-lists obtained from the rental service company Netflix, which included 18,000 titles watched by over 480,000 users [Bennett and Lanning, 2007]. The watch-lists were analysed to identify pairs of movies watched by the same persons within a range of five days and both rated 5 out of 5 stars. Neither the order nor the distance between the movies were considered as relevant, corresponding to the parameters set to $\delta = 5$ (maximum distance in days), $\alpha_J = 0.2$ (distance not influent), $\gamma = 0.5$ (order not influent). The popularity bias parameter was set to $\beta = 0.9$ to strongly punish over-popular movies in the watch-lists.

The result of the analysis was the compilation of lists of 'top associated titles' according to Netflix users. Each list reports titles loved by people who also rented and loved the 'seed' movie within a short period. Two sample lists of these 'top associated titles' are reported hereafter.

Sense and sensibility (1995). This movie was watched by 31,389 Netflix users. Within a period of five days, the same users also watched and equally rated 8,966 other movies. The top associated movies uncovered were: 'Shakespeare in Love' (1998), 'A Fish Called Wanda' (1988), 'Amadeus' (1984), 'The Full Monty' (1997) 'Four Weddings and a Funeral' (1994), 'Elizabeth' (1998), 'Field of Dreams' (1989), 'Much Ado About Nothing' (1993), 'The Princess Bride' (1987), 'Dead Poets Society' (1989).

Jumanji (1995). This movie was watched by 15,078 Netflix users. Within the same five days, these users also watched and assigned the same rating assigned

to ‘Jumanji’ to 7,406 other movies. The top associated movies turned out to be: ‘Hook’ (1991), ‘Kindergarten Cop’ (1990), ‘Twister’ (1996), ‘Flubber’ (1997), ‘Beetlejuice’ (1988), ‘Liar Liar’ (1997), ‘Men in Black’ (1997), ‘City Slickers’ (1991), ‘A League of Their Own’ (1992), ‘Father of the Bride’ (1991).

Monografies de l'Institut d'Investigació en Intel·ligència Artificial

Num. 1	J. Puyol, *MILORD II: A Language for Knowledge–Based Systems*
Num. 2	J. Levy, *The Calculus of Refinements, a Formal Specification Model Based on Inclusions*
Num. 3	Ll. Vila, *On Temporal Representation and Reasoning in Knowledge–Based Systems*
Num. 4	M. Domingo, *An Expert System Architecture for Identification in Biology*
Num. 5	E. Armengol, *A Framework for Integrating Learning and Problem Solving*
Num. 6	J. Ll. Arcos, *The Noos Representation Language*
Num. 7	J. Larrosa, *Algorithms and Heuristics for Total and Partial Constraint Satisfaction*
Num. 8	P. Noriega, *Agent Mediated Auctions: The Fishmarket Metaphor*
Num. 9	F. Manyà, *Proof Procedures for Multiple-Valued Propositional Logics*
Num. 10	W. M. Schorlemmer, *On Specifying and Reasoning with Special Relations*
Num. 11	M. López-Sánchez, *Approaches to Map Generation by means of Collaborative Autonomous Robots*
Num. 12	D. Robertson, *Pragmatics in the Synthesis of Logic Programs*
Num. 13	P. Faratin, *Automated Service Negotiation between Autonomous Computational Agents*
Num. 14	J. A. Rodríguez, *On the Design and Construction of Agent-mediated Electronis Institutions*
Num. 15	T. Alsinet, *Logic Programming with Fuzzy Unification and Imprecise Constants: Possibilistic Semantics and Automated Deduction*
Num. 16	A. Zapico, *On Axiomatic Foundations for Qualitative Decision Theory - A Possibilistic Approach*
Num. 17	A. Valls, *ClusDM: A multiple criteria decision method for heterogeneous data sets*
Num. 18	D. Busquets, *A Multiagent Approach to Qualitative Navigation in Robotics*
Num. 19	M. Esteva, *Electronic Institutions: from specification to development*
Num. 20	J. Sabater, *Trust and Reputation for Agent Societies*

Num. 21 J. Cerquides, *Improving Algorithms for Learning Bayesian Network Classifiers*

Num. 22 M. Villaret, *On Some Variants of Second-Order Unification*

Num. 23 M. Gómez, *Open, Reusable and Configurable Multi-Agent Systems: A Knowledge Modelling Approach*

Num. 24 S. Ramchurn, *Multi-Agent Negotiation Using Trust and Persuasion*

Num. 25 S. Ontañon, *Ensemble Case-Based Learning for Multi-Agent Systems*

Num. 26 M. Sánchez, *Contributions to Search and Inference Algorithms for CSP and Weighted CSP*

Num. 27 C. Noguera, *Algebraic Study of Axiomatic Extensions of Triangular Norm Based Fuzzy Logics*

Num. 28 E. Marchioni, *Functional Definability Issues in Logics Based on Triangular Norms*

Num. 29 M. Grachten, *Expressivity-Aware Tempo Transformations of Music Performances Using Case Based Reasoning*

Num. 30 I. Brito, *Distributed Constraint Satisfaction*

Num. 31 E. Altamirano, *On Non-clausal Horn-like Satisfiability Problems*

Num. 32 A. Giovannucci, *Computationally Manageable Combinatorial Auctions for Supply Chain Automation*

Num. 33 R. Ros, *Action Selection in Cooperative Robot Soccer using Case-Based Reasoning*

Num. 34 A. García-Cerdaña, *On some Implication-free Fragments of Substructural and Fuzzy Logics*

Num. 35 A. García-Camino, *Normative Regulation of Open Multi-agent Systems*

Num. 36 A. Ramisa Ayats, *Localization and Object Recognition for Mobile Robots*

Num. 37 C.G. Baccigalupo, *Poolcasting: an intelligent technique to customise music programmes for their audience*

Num. 38 J. Planes, *Design and Implementation of Exact MAX-SAT Solvers*

Num. 39 A. Bogdanovych, *Virtual Institutions*

Num. 40 J. Nin, *Contributions to Record Linkage for Disclosure Risk Assessment*

Num. 41 J. Argelich Romà, *Max-SAT Formalisms with Hard and Soft Constraints*

Num. 42 A. Casali, *On Intentional and Social Agents with Graded Attitudes*

Num. 43 A. Perreau de Pinnick Bas, *Decentralised Enforcement in Multiagent Networks*