2014-05-05

A Model-Based Music Recommendation System for Individual Users and Implicit User Groups

Yajie Hu
University of Miami, huyajiecn@hotmail.com

UNIVERSITY OF MIAMI

A MODEL-BASED MUSIC RECOMMENDATION SYSTEM FOR INDIVIDUAL USERS AND IMPLICIT USER GROUPS

By

Yajie Hu

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

May 2014

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

A MODEL-BASED MUSIC RECOMMENDATION SYSTEM FOR INDIVIDUAL USERS AND IMPLICIT USER GROUPS

Yajie Hu

Approved:

HU, YAJIE (Ph.D., Computer Science)
(May 2014)

A Model-based Music Recommendation System
for Individual Users And Implicit User Groups

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Mitsunori Ogihara
No. of pages in text: (103)

This dissertation introduces several problems on music recommendation. To make a high-qualified recommender, we evaluate feature importance for favorite song detection from two perspectives. The experiment results expose that collaborative filtering signal is the most important feature among the analyzed features. A classifier combination method is proposed to leverage several classifiers trained by different data sources to predict music genre. The complemented genre labels are used in a recommendation system for individual users on local device. The recommender takes freshness, time pattern, genre, publish year, and favor into account to make recommendations. The recommender outperforms the baseline on mostly favorite songs. We propose an adapted recommendation method to response user feedbacks and find out local optimizations to improve the recommendation quality. Furthermore, a probability-based method is proposed to make recommendations for implicit user groups by integrating individual opinions on music.

ACKNOWLEDGMENTS

When I finished my dissertation, I suddenly realized that my doctoral program was almost finished. The dream to be a doctor of philosophy is about to be true, but the dream sounded incredible four years ago. Although the acknowledgement may be not read by many people, I still would like to use this page to give the most sincere gratitude to my parents, my advisor, and anyone who supports and encourages me in this period.

My parents encouraged me to research on this topic and pursue the Ph.D. degree, even though I had to leave from them for a long time. Additionally, the love given by my parents made me overcome tough times and warmed my heart. Thanks very much for them!

I am very lucky and honored to be advised by Professor Ogihara, Mitsunori. He strongly supported my research and generously share his ideas and opinions. Furthermore, he allowed me to control my time and progress by myself, so I had enough time to attempt new methods and improve them. All achievements in my doctoral program could not be obtained without my advisor's supports. Nothing is better than big thanks.

Thanks a lot to my friends and roommates. They gave me many impressive moments and made my life colorful.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivations

In recent decades, technologies on music storage and distribution dramatically changed typical music consumption behavior. Personal music collections in pockets have rocketed so that the scale of available tracks in hands has grown from tens to thousands. Technological improvements on portable storage make storing thousands of tracks on device possible. High speed network combined with online music services enables huge music collections of music providers accessible for users as long as the devices connect the network and the services are subscribed.

However, a great number of songs have de-emphasised their value. It is common that users own so many digital music on device that most of them are played once, or not at all. Similarly, music providers pay a lot for the copyright of huge music collection, but only a small subset are recommended to users or the recommended songs are not attractive. It is obviously desirable that effective recommendation approaches are proposed to create personalized playlists, and make hidden "treasure" as rich as it is.

So far, music search engines allow users to search songs by metadata, social tags (labels) or even humming. In some cases, there is however no metadata, or only file names. Only audio content is not enough to effectively browse or search them. Incorrect tags (labels) might mislead music search engines into wrongly feeding so that the results confuse users. Even though metadata of tracks is completed and corrected by the manually labeled tracks matched by a music fingerprint system, and the social tags are cleaned, it is usually the case that users have no idea about what to search just for a rest, or they are tired to search music to listen.

Although structured information is able to represent music for an information retrieval system, some other retrieval tasks are impossible to complete. For example, users would like to listen to songs which are similar to their favorite songs in terms of genre, melody, pitch and/or more. Furthermore, the relevance may be the combination of different types of conditions, like acoustic similarity, social tags, and listening context. A classical retrieval system is hardly able to make a recommendation as good as a recommender could be. In addition, recommendation results assist user to discovery largely unexplored and potentially interesting music.

For a group of users in a social event, or an arbitrary formed group, how to recommend songs for the group could be solved only by a music recommender for user groups instead of a music search engine. Good recommendation results for groups in social events would help an atmosphere of delight, and contribute to find common favor or even build up a musical relationship in an arbitrary formed group.

1.1.1 Industry

Many music recommendation systems for online music services emerged in recent years, such as *Pandora*, *Last.fm*, *Amazon*, *Google Play Music*, *iTunes* and so forth. The provided recommendations are the critical core to retain users, and the quality of music recommendation results from the majority of the product sales. Greg Linden, who implemented the first recommendation engine for *Amazon*, states[1]:

(Amazon.com) recommendations generated a couple orders of magnitude more sales than just showing top sellers.

Online radio stations

Many online radio station providers, like *Pandora*[2], *Spotify*[3] and *douban radio*[4], enable users to create a radio station starting with a song, an artist or a tag, and the recommenders recommend a song after another. It is perfect if users don't know what to listen to, or if users would like to explore something new. Similarly, the recommenders of *Google Play Music*[5] and *iTunes Genius*[6] automatic generate a playlist, which is static no matter that user's listening behavior is skipping songs at the beginning or playing the whole song.

wah-wah[7] provides new service: "co-listening". This service allows a user to create a personal radio station with an activity label, and enables the user broadcast the radio station in public. Other users can filter public radio stations by activities or other attributes, and enter a station to listen to music together. It allows users to enjoy music with friend or strangers doing the same activity. However, there is no proof that the recommender takes the profiles of all group

[1]http://glinden.blogspot.com/2007/05/google-news-personalizationpaper.html
[2]http://www.pandora.com/
[3]https://www.spotify.com/
[4]http://douban.fm/
[5]https://play.google.com/music/
[6]http://www.apple.com/itunes/
[7]http://app.wahwah.co/

members into account to bring targeted tracks to the group, so it might merely provide a platform to share music recommended to the host with other guests.

Online music stores

As all e-commerce sites, online music stores sell music by songs or albums online, like online book or movie stores. One of most important features for music stores is recommending potentially interesting products, which are relevant to the favorite or purchased products, like "*Recommendations for You in MP3 Downloads*" at *Amazon*. Because of the demand, collaborative filtering techniques are researched deeply, and widely leveraged in industry. The recommendation results are sometimes explicitly shown with the description: "*Customers Who Bought This Item Also Bought*".

Online music stores consider tracks or albums as products, so the context information, including recently played songs, current activities, mood, and so forth, are not taken into account.

1.1.2 Academia

Since 2001, research in music recommendation has expanded as the digital music became popular. Figure 1 presents the number of papers[8] related to music recommendation since 2001, and it implies the increasing interest in this field. The data of the figure is collected from *Google Scholar*[9]. For each year, we count the number of results from *Google Scholar* which contain the terms, "music

[8]Indexed by Google Scholar September 12th, 2013.
[9]http://scholar.google.com/

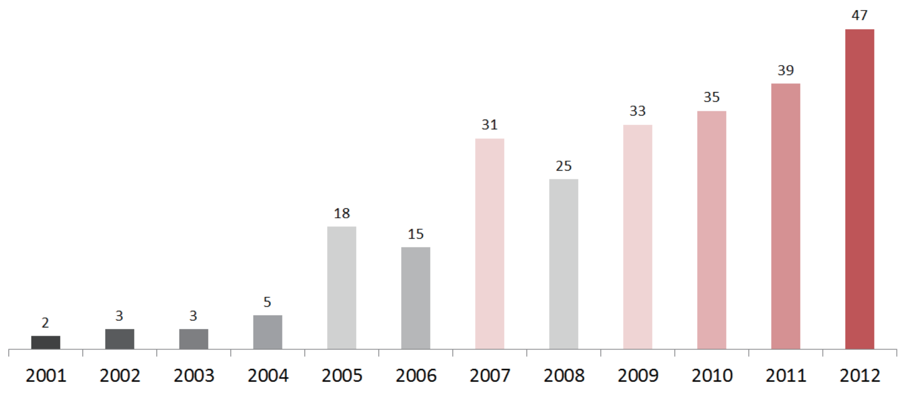


Figure 1: Number of academic papers related to music recommendation.

recommendation" or "music recommender", in the title of the paper.

A closer look at the Music Information Retrieval (MIR) society, which researches music recommendation and related problems from different views, shows a remarkable increasing interest in music recommendation and discovery. Table 1 summarizes related paper by topics, published in all ISMIR (International Society for Music Information Retrieval[10]) conference.

1.2 What's an ideal music recommender?

When a user has a great number of choices of music to listen to, like browsing a non-personalized music catalog, the user is overwhelmed by the choices. A recommender is needed to filter songs and find new possibly favored songs to the user. The recommender should maximize the user's satisfaction by playing appropriate song at the right time, and, meanwhile, minimize the user's effort on starting the recommender and providing feedback.

The recommendation problem can be seen as a ranking problem. For each recommendation, a recommender creates a sorted list ordering by suitability of

[10]http://www.ismir.net/

Topics	References
Content-based recommendation	[Logan, 2002], [Logan, 2004]
User profiling aspects	[Chai and Vercoe, 2000], [Uitdenbogerd and Schnydel, 2002]
Prototype systems	[Celma et al., 2005], [Gulik and Vignoli, 2005], [Pampalk and Goto, 2007], [Hu and Ogihara, 2011]
Playlist generation	[Pauws and Eggen, 2002], [Pampalk et al., 2005], [Pauws and Wijdeven, 2005], [Pampalk and Gasser, 2006], [Pauws et al., 2006], [Oliver and Kreger-Stickles, 2006], [Masahiro et al., 2008], [Flexer et al., 2008], [Maillet et al., 2009], [Bosteels et al., 2009], [McFee and Lanckriet, 2011], [Lee et al., 2011], [Moore et al., 2012]
Social tagging	[Baccigalupo et al., 2008], [Symeonidis et al., 2008]
Graphics user interface	[Donaldson, 2007b], [Miller et al., 2010]
Music similarity	[Anglade et al., 2007], [Fields et al., 2008], [Seyerlehner et al., 2009], [McFee and Lanckriet, 2009], [McFee et al., 2010], [Mak et al., 2010]
Hybrid recommendation	[Yoshii et al., 2006], [Tiemann and Pauws, 2007], [Yoshii et al., 2007], [Magno and Sable, 2008], [Yoshii and Goto, 2009], [Schedl and Flexer, 2012]
Sociological aspects	[Cunningham et al., 2006], [McEnnis and Cunningham, 2007], [Chordia et al., 2008], [Cunningham and Nichols, 2009], [Barrington et al., 2009], [Laplante, 2011]
Feature analysis	[Hu et al., 2006], [Bogdanov and Herrera, 2011]
Recommendation behavior survey	[Kamalzadeh et al., 2012], [Lee and Waterman, 2012]

Table 1: Papers related to music recommendation in ISMIR conferences.

songs at that moment, and the top songs are selected as the recommendation results. The ideal recommender is supposed to recommend songs to hit the top songs of the user's favorite list, each time.

1.2.1 Difference to movies and books recommenders

Since music recommendation is a special field of recommendation problem, some music recommendation systems apply or adapt existing recommendation approaches, such as collaborative filtering, into the music domain. In some music recommendation systems, the relevance between two items is evaluated by analyzing music content to recommend items which are similar to a favored item. A music recommender has obvious difference to a recommender for books or movies, but the outer framework, i.e., how to leverage the relevance for recommendation, is similar to movies and books recommenders.

Music is somewhat different from other domains. It is common that a user can play an item (e.g., a track, an album or a playlist) several times, even continuously and repeatedly. Listening to music could be more relaxing than watching movies or reading books. Users might do other things when listening to music, so it is impossible to track users' preferences by explicit and instant ratings or comments. Implicit listening behavior, i.e., skipping or finishing a song, is sometimes the only available way to collect users' feedback. In addition, the average duration of a song is about 4 minutes, so a user usually consumes many items in a listening session. The recommender is possible to track and adapt users' preferences using the implicit feedback on previous songs. Furthermore, in the session, the user might be bored to the current genre, and migrate to another genre. The system needs to recognize and track listening patterns and adapt to these changes.

The listening context is another significant difference between music and other two domains. In different contexts, a user tends to consume different music. The context includes activity context and listening session context. For instance, dance/electronic contributes to create atmosphere in a party, blues is better while people are having dinner, and classical piano suitably follows opera.

1.2.2 Confidence and novelty

A recommender longs for accurate predictions which people will would like to listen to. However, accurate predictions are not what a good recommender is all about. If a recommender brings the most popular songs to users, it might be successful in predicting moderately liked songs, but the recommendation is non-personalized, boring, and not useful for users to discover more new and more interesting songs. One of potential goals for a user using a music recommender may be to find songs the user was not aware of which the user would consider a favorite. Novelty is an important quality in the recommendations of a recommender, but novelty cannot be achieved unless the system understands the nature of risk in a recommendation, and can be programmed to take risks.

However, providing something popular and good is not all wrong. While recommending a popular song might make the recommendation seem generic, they can nevertheless give a feeling of confidence in the system. Building up the confidence would keep listener using the recommender and tolerating a few bad recommendations. Therefore, a good recommender is supposed to keep a balance between confidence and novelty.

1.2.3 Listening context

Listening context has an influence on users' choices, so it should be considered as a factor in music recommendation systems. For example, a user like energetic music while doing exercises, and love relaxing music while working or having a rest. The current choices tightly depend on what the user is doing. Even though a recommender plays the user's favorite classical music, it can't make the user satisfied if the user is in gym.

The session context is also the factor which would influences on the recommendations. Since a song is not as long as a movie, people usually listen to several songs. In a session, some listeners would like to stay in a similar genre or even one artist, whereas some users prefer to mixing music in different genre. Tracking listening context may assist to recognize the listening pattern and predict the next one or several songs by the pattern.

1.2.4 Adaptive personalized recommendation

Even though a well-trained recommender is able to make very good recommendations, it it cannot guarantee that every recommendation is loved. If a recommended song is skipped in a few seconds, i.e., the implicit feedback is negative, the recommender is supposed to respond to the feedback. The computational complexity of retraining recommends makes retraining impractical, so an adaptive and fast approach for response to negative feedback is necessary. For example, re-ranking possible recommendations by personalized ranking methods can adjust the recommendation results.

1.2.5 Feed for a group of users

Music can be shared with friend or strangers together. For example, friends in a party and customers in a restaurant listen to music together. In addition, some apps, like *wahwah*, support the formation of user communities around user-created on-line radio stations, and listen to music together. Members of the community whose tastes are too far from the majority taste, can easily leave the community, thus refining the focus of community. The recommender for user groups is expected to recommend songs which make the majority of a group comfortable and stratified, since it would be impossible that the recommendation results cater to all members' taste.

1.3 Our proposal

In this dissertation, we considered the aforementioned factors for being a good recommender, and proposed music recommendation algorithms and solved related problems. The dissertation is made up of four parts. The four topics covered each in a separate chapter of the dissertation can be integrated in the a recommendation system. To make each part clear, we will introduce and evaluate them part by part.

1. Feature importance evaluation for music recommendation

Acoustic features, collaborative filtering signal, and social tags could represent songs and be used to rank songs in music recommendation systems, but we don't know which one works and how much they are important to make recommendations. Thus, we evaluate and compare different types of features from two perspectives in terms of the importance for music recommendation. We conclude that collaborative filtering signal outperforms over other considered features.

2. A recommendation system for individual users

   A music recommendation algorithm is proposed to recommend music for individual users on their devices. The proposed algorithm emphasizes novelty and confidence, considers listening patterns, and leverages forgetting curve, time series analysis, and listening time distribution to rank possible songs. The top song in the ranked list is recommended and played, when the current song is finished or skipped.

   As a high level acoustic feature, genre is used in the proposed recommender, but the metadata of songs, like ID3v1 or ID3v2 header, sometimes misses genre data. We therefore proposed a method to integrate different types of data sources, i.e., acoustic signal, lyrics, artist terms and social tags, to predict song genre. The method combines several classifiers, which are trained by one data source among them, by confidence and authority.

3. Adaptive method for personalized recommendation

The recommendation problem is treated as a ranking problem in the view of information retrieval, and music is represented by metadata of songs and compressed collaborative filtering signal. Coordinate descent is applied to minimize the ranking cost function. The features are grouped according to types. With theoretical speed-up, the training process is so fast that it is sufficient to be able to include real-time the user's feedback to improve the recommendation quality.

4. A recommendation system for implicit user groups

   In some cases, people listen to music together. They maybe listen together in a party or a dinner, or in a public online radio station. Music recommenders for user groups are supposed to make as many members as possible satisfied, but it may be impossible to make every member happy.

   We propose a method to predict the probability which the member of a group stays in the group, and rank songs by the probability combined with the ratings of these songs by the member. The probability is predicted using the listening log of the members and how similar they are in terms of the music preference. In addition, the staying probability might be used to recommend groups to a new user.

1.4 Summary of contributions

The main contributions of this dissertation are:

1. Feature importance evaluation for favorite songs detection.

   We evaluate feature importance from two perspectives. One is to consider favorite songs detection as a classification problem, and the other is to evaluate similarity metrics for detecting favorite songs. We apply feature selection methods, which are commonly used in the preprocessing of classification systems, and statistical methods to evaluate similarity metrics on different features.

2. Genre classification using confidence-based classifiers combination.

   Since songs have several data types, we trained sub-classifiers by different types of data. These sub-classifiers are combined using both classifier authority and classification confidence for a particular instance. In the experiments, the combined classifier surpasses all of these sub-classifiers and the SVM classifier using concatenated vectors from all data types.

3. A music recommendation system for local music collection.

   The proposed music recommendation algorithm has four new points: Breaking the assumption that the next song must be similar to the current song; Emphasis on novelty using forgetting curve; considering the time pattern of playing behaviors; and dynamic weights of factors to recommend the next song.

4. A fast learning to rank approach for adaptive personalized recommenders.

   We propose a method to represent music to include metadata of songs and collaborative filtering signal. The ranking algorithm groups features in order to speed up the training process. The experimental results show that the proposed approach trains rankers faster than other learning to rank approaches and the ranking results of the proposed approach outperforms others as well.

5. Music recommendation for implicit user groups.

   We predict every member's rating to a song using a probabilistic model. Then, the individual ratings are aggregated by the staying probability, which is estimated by how much the user's taste is similar to others.

1.5 Dissertation outline

Chapter 2 categorizes and introduces popular recommendation methods in academia and systems in industry. The category of state-of-the-art research presents the mainstreams of this field. The various recommendation modes show the variety of recommendation challenge, and we announce the mode which we focus on in this dissertation. Available open dataset are introduced and compared at the end of this chapter.

We evaluate several kinds of features in terms of importance for music recommendation in Chapter 3. These features cover acoustic content, musical attributes, artist attributes, and user-item relationship. We evaluate these features from two perspectives, i.e., correlation of music similarity according to a particular feature, and feature selection in statistics. The experimental results are interesting. For example, song hotness is not as important as we intuitively expect.

Chapter 4 presents the proposed recommendation method and song genre classification approach. The recommendation method estimates the probability of a song to be recommended from five perspectives: song genre, publish year, freshness, favor and time pattern. These factors are integrated by a dynamic adapted algorithm. Because the genre tag of a song file is sometimes invalid, an automatically genre classifier is in need to complement these genre labels. This classification method applies several sub-classifiers to deal with different types of the data source, and integrates the results of these sub-classifiers to predict genre labels.

Chapter 5 introduces a new method to fast learn a user's feedbacks, and adaptively recommend music to the user. The recommendation problem is solved as a ranking problem, so a fast learning to rank method is proposed. Furthermore, the collaborative filtering signal is compressed and represented as a vector to train the ranking model. The experimental results show that the proposed method is fast enough to train the ranking model online. The recommendation quality is better than that of the memory-base collaborative filtering technique and most of learning to rank methods.

We discuss music recommendation for user groups in Chapter 6. The individual opinions of group members are estimated by a collaborative filtering model. These opinions are integrated by the probability that the members stay in the group for the next recommendation. Essentially, the recommendations for the group are expected to make the majority satisfied.

Chapter 7 summarizes the recommendation method for individual users and user groups.

Chapter 2

Related Work

Nowadays, many successful music recommendation systems have been built as typical e-Commerce systems or personalized radio stations, and they help people choose and purchase from countless items or enjoy music online. The potential profitable market attracts a lot of attentions from industry, and a great deal of emphasis on music recommendation is placed in academic research as well. Researchers investigate different user cases, from just browsing, finding good items, to finding trustable recommenders and generating playlists. Many popular approaches in general recommendation domains are used used in the music recommendation domain with more or less modifications, whereas some novel algorithms are proposed to solve music recommendation problems. We categorize these approaches in several classes.

2.1 Recommendation methods

2.1.1 Content-based filtering

A content-based filtering recommender extracts relevant features to describe items, and predict which items the user would like based on the user's profile. The predictions made by content-based filtering do not rely on other user ratings but on the content of the items. It recommends songs similar to the favorite songs, and removes songs that are similar to the skipped songs. The key part of this approach is the similarity function among items.

In an approach proposed by Cano et al. [Cano et al., 2005], acoustic features of songs are extracted and used, such as timbre, tempo, meter and rhythm patterns. Furthermore, some work expresses similarity according to the song's emotional content. Cai et al. [Cai et al., 2007] recommends music based only on emotion. Logan [Logan, 2004] tried four similarity functions to calculate the distance from a song to a song set with the same acoustic features. Artist similarity is analyzed by the integration of several heterogeneous data sources, i.e., social, semantic and acoustic features [McFee and Lanckriet, 2009]. Mak et al. [Mak et al., 2010] present a similarity function to correlate with human perception using several acoustic features.

Content-based filtering has several shortcomings:

- When a new user enters to the system, the recommender suffers from the cold-start problem due to the lack of the user's preference information.
- If the similarity function works well to find out the similar songs to a user's preferences, the user will always receive the same items similar to the user's favorite songs, so the recommender cannot give novel recommendations.
- Songs are described by selected features related to song content, so the recommender performance is limited by the performance of the feature extraction component.
- Similarity functions measure the similarity of machine recognizable features of the content, so the user opinion (subjectivity) is not taken into consider.

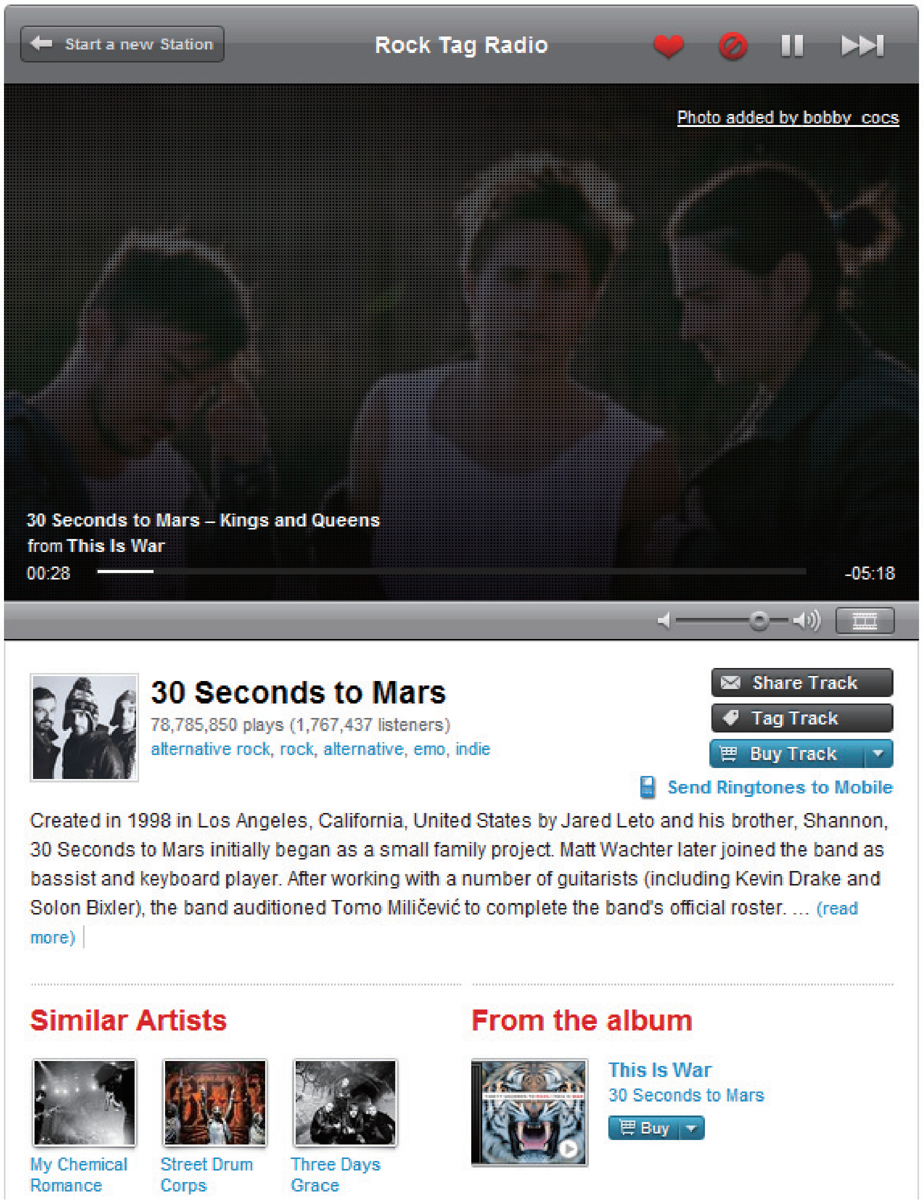
2.1.2 Collaborative filtering

Collaborative filtering approach collects user-item ratings data, and predicts user preferences by the user-item relationship. This approach is widely used in many domains, and it is popular in music recommendation as well. In general, collaborative filtering has three types, i.e., memory-based filtering, model-based filtering, and hybrid filtering. The memory-based filtering exploits the similarity among items or users based on ratings data, and makes recommendations. The typical examples are neighborhood based collaborative filtering and item-based/user-based top-N recommendations. Using data mining, machine learning methods, model-based methods train recommenders on the training data by finding rating patterns, and the trained recommenders are used to recommend items to users. Hybrid methods combine memory-based filtering and model-based filtering.

Using collaborative filtering, music recommenders bring pieces of music to a user based on rating of those pieces by other users with similar taste [Cohen and Fan, 2000]. McFee et al. [McFee et al., 2010] present a method to improve content-based recommendation in the long tail by learning a similarity function from collaborative filtering data. This principle has been used by various social websites, including Last.fm (Figure 2), myStrands.

Collaborative filtering presents some limitations:

- Cold-start problem occurs when a new user or item enters the recommendation system, because the recommender is short of the ratings by the new user or on the new item. It is tough to recommend the item without ratings.



30 Seconds to Mars
78,785,850 plays (1,767,437 listeners)
alternative rock, rock, alternative, emo, indie

Created in 1998 in Los Angeles, California, United States by Jared Leto and his brother, Shannon, 30 Seconds to Mars initially began as a small family project. Matt Wachter later joined the band as bassist and keyboard player. After working with a number of guitarists (including Kevin Drake and Solon Bixler), the band auditioned Tomo Miličević to complete the band's official roster. ... (read more)

Figure 2: Last.fm recommendation system.

- As the song collection goes huge, data sparsity appears since a user might rate only 1% or fewer songs in the huge collection. Data sparsity makes recommenders be hard to find reliable neighbors.

- If a user's taste is so unusual that the ratings on the user's favorite songs are very a few, the user will not have many neighbors. As a result, the recommendations would be poor.

- The early ratings initialize the recommendation system, and the current recommendations are made based on these early ratings. In turn, the current recommendations receive feedbacks, and are recommended then. Thus, the early ratings make too much effect on the recommendation system.

- Popular songs which have many ratings are likely to be recommended, and will receive more ratings. As a result, the rich get richer. The popular songs overwhelm new or unpopular songs, so the novelty and personalization would be weakened.

2.1.3 Context-based filtering

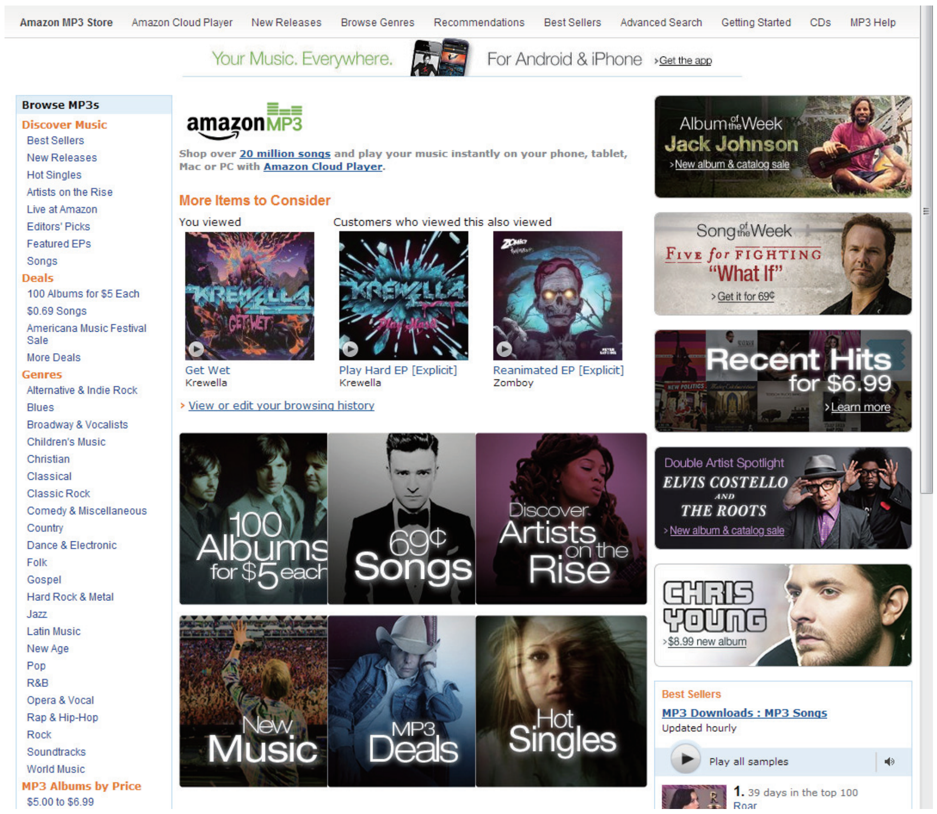
Context information is defined as the environment around the user while listening to music. It includes physical location, current mood, activity, what songs the user has played, and so forth. Compared to other entertainment domains, music is more relaxing, since users are able to do something while listening to music. Consequently, music recommendation obviously depends on the current environment. Furthermore, users usually listen to more than one song, so the previous song would influence the current choice. In addition, in the listening session, the user's interest might change due to the context change.

Liu et al. [Liu et al., 2009] take the change in the interests of users over time into consideration, and add time scheduling to the music playlist. Su et al. [Su and Yeh, 2010] improve collaborative filtering using user grouping by context information, such as location, motion, calendar, environment conditions and health conditions, while content analysis assists system to select appropriate songs.

2.1.4 Hybrid methods

Combination music content and other information is receiving more attention lately. The goal is to achieve a better and more reliable recommendation system by improving the shortcomings of each system when used separately. Two or more recommendation methods are combined by weighted, switching, mixed, cascade, and probability model.

Donaldson [Donaldson, 2007a] leverages both spectral graph properties of an item-base collaborative filtering as well as acoustic features of the music signal. Shao et al. [Shao et al., 2009] use both content features and user access pattern to recommend music. Yoshii et al. presents two hybrid methods to combine content-based recommendation method and collaborative filtering method. One method [Yoshii et al., 2006] uses Bayesian network to integrate both ratings and content data. [Yoshii and Goto, 2009] proposes an extended probabilistic latent semantic indexing, which incorporates Gaussian mixture models, to combine ratings and content data.



Shop over 20 million songs and play your music instantly on your phone, tablet, Mac or PC with Amazon Cloud Player.

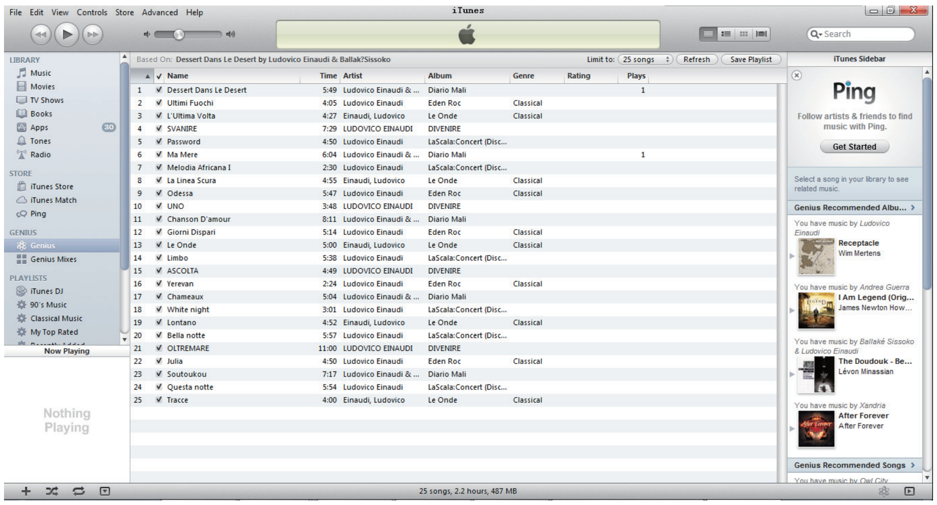
Figure 3: Music recommendation for browsing in Amazon MP3.

2.2 Recommendation modes

2.2.1 Browsing items

In online music store, music recommenders are expected to show a user's favorite songs or albums when the user is browsing music in the store. The recommenders have to follow the category in the store and recommend relevant and interesting items. In addition, the songs which are similar to the purchased songs are supposed to be provided for browsing.

Amazon MP3, *Google Play Music*, and *iTunes* are popular online music stores. All of them provide a platform to assist users to browse music and discover what they are interested in, as shown in Figure 3.



Based On: Dessert Dans Le Desert by Ludovico Einaudi & Ballak?Sissoko

	✓	Name	Time	Artist	Album	Genre	Rating	Plays
1	✓	Dessert Dans Le Desert	5:49	Ludovico Einaudi & ...	Diario Mali			1
2	✓	Ultimi Fuochi	4:05	Ludovico Einaudi	Eden Roc	Classical		
3	✓	L'Ultima Volta	4:27	Einaudi, Ludovico	Le Onde	Classical		
4	✓	SVANIRE	7:29	LUDOVICO EINAUDI	DIVENIRE			
5	✓	Password	4:50	Ludovico Einaudi	LaScala:Concert (Disc...			
6	✓	Ma Mere	6:04	Ludovico Einaudi & ...	Diario Mali			1
7	✓	Melodia Africana I	2:30	Ludovico Einaudi	LaScala:Concert (Disc...			
8	✓	La Linea Scura	4:55	Einaudi, Ludovico	Le Onde	Classical		
9	✓	Odessa	5:47	Ludovico Einaudi	Eden Roc	Classical		
10	✓	UNO	3:48	LUDOVICO EINAUDI	DIVENIRE			
11	✓	Chanson D'amour	8:11	Ludovico Einaudi & ...	Diario Mali			
12	✓	Giorni Dispari	5:14	Ludovico Einaudi	Eden Roc	Classical		
13	✓	Le Onde	5:00	Einaudi, Ludovico	Le Onde	Classical		
14	✓	Limbo	5:38	Ludovico Einaudi	LaScala:Concert (Disc...			
15	✓	ASCOLTA	4:49	LUDOVICO EINAUDI	DIVENIRE			
16	✓	Yerevan	2:24	Ludovico Einaudi	Eden Roc	Classical		
17	✓	Chameaux	5:04	Ludovico Einaudi & ...	Diario Mali			
18	✓	White night	3:01	Ludovico Einaudi	LaScala:Concert (Disc...			
19	✓	Lontano	4:52	Einaudi, Ludovico	Le Onde	Classical		
20	✓	Bella notte	5:57	Ludovico Einaudi	LaScala:Concert (Disc...			
21	✓	OLTREMARE	11:00	LUDOVICO EINAUDI	DIVENIRE			
22	✓	Julia	4:50	Ludovico Einaudi	Eden Roc	Classical		
23	✓	Soutoukou	7:17	Ludovico Einaudi & ...	Diario Mali			
24	✓	Questa notte	5:54	Ludovico Einaudi	LaScala:Concert (Disc...			
25	✓	Tracce	4:00	Einaudi, Ludovico	Le Onde	Classical		

25 songs, 2.2 hours, 487 MB

Follow artists & friends to find music with Ping.

Select a song in your library to see related music.

You have music by *Ludovico Einaudi*
Receptacle
Wim Mertens

You have music by *Andrea Guerra*
I Am Legend (Orig...
James Newton How...

You have music by *Ballaké Sissoko & Ludovico Einaudi*
The Doudouk - Be...
Lévon Minassian

You have music by *Xandria*
After Forever
After Forever

You have music by *Owl City*

Figure 4: Genius recommendation system in iTunes.

2.2.2 Automatic playlist generation

This mode focuses on recommending songs that are similar to chosen seed songs, and the generation of playlist. Ragno et al. [Ragno et al., 2005] recommended an approach to recommend music that is similar to chosen seeds as a playlist. Similarly, Flexer et al. [Flexer et al., 2008] provided a sequence of songs to form a smooth transition from the start to the end. These approaches ignore user's feedback when the user listens to the songs in the playlist. All seed-based approaches have the underlying problem that they produce excessively uniform lists of songs if the dataset contains lots of music cliques.

iTunes Genius employs similar methods to generate a playlist from a seed as shown in Figure 4. Online radio stations provided by *Google Music All Access* generates self-extended playlist as well.

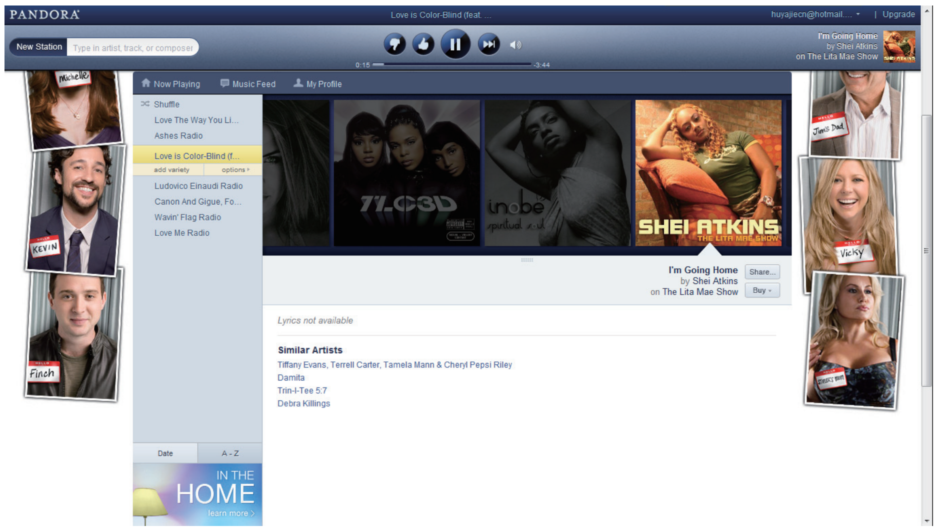


Figure 5: Pandora recommendation system.

2.2.3 *Dynamic music recommendation*

Dynamic music recommendation improves automatic playlist generation by considering the user's feedback. In the method proposed by Pampalk et al. [Pampalk et al., 2005], playlist generation starts with an arbitrary song and adjusts the recommendation result based on user feedback. This type of method is similar to *Pandora* shown in Figure 5.

2.2.4 *Recommendation for user groups*

Sharing songs with friends and strangers is fun, and this sharing contributes to an exciting atmosphere at a party or dinner. MusicFX [McCarthy and Anagnost, 1998] recommends music station to play in a fitness center. The system needs the members in the gym to rate all the stations beforehand, and the station with the highest average rating is played. The shortcoming that users must rate all of the stations makes MusicFX difficult to scale to a large number of possible choices. Flytrap [Crossen et al., 2002] learns user preferences from the listening history, and selects songs which are similar to the favorite songs in respect of artist and genre to play in a public place. Adaptive Radio [Chao et al., 2005] makes the group recommendation from an opposite view. It collects all negative preferences of the group members, and finds out the similar songs of these dislike songs. After filtering out the dislike songs and the similar ones, the remained songs are recommended to the group. One of its shortcomings is that if the group has a *gray sheep* who disagrees with the other users, the favorite songs for all except the gray sheep would be removed out. [Anglade et al., 2007] discusses how to create music channels for visual groups using graph-based community extraction techniques.

wahwah provides an iOS app to support "co-listening" such that users can join or leave user-created public radio stations. The service lets users generate a group and listen to music together.

2.3 Available dataset

There are three music user log datasets available online, i.e., Taste Profile Dataset[1] by the Echo Nest, Last.fm dataset 360K[2] and Yahoo! Music User Ratings of Songs[3] by Yahoo! Research.

The Taste Profile Dataset is a huge collection of real world anonymous listener data in the form of Echo Nest Taste Profiles. Considering the protection of individuals private information, the data includes a shuffled hash of persistent session identifiers from a very small random selection of the musical universe and only

[1]http://labrosa.ee.columbia.edu/millionsong/tasteprofile
[2]http://mtg.upf.edu/node/1671
[3]http://webscope.sandbox.yahoo.com/catalog.php?datatype=r

play counts associated with Echo Nest song IDs that overlap with the Million Song Dataset (MSD)[4]. No usernames, listener details, original IDs, dates, IPs, locations or anything but random user string, namely, Echo Nest song ID, and play count are released[5]. The Taste Profile dataset has 1,019,318 unique users, 384,546 unique MSD songs and 48,373,586 <*user ID*, *song ID*, *play count*> triplets. The triplets don't have any time stamps, and they are not in chronological order. However, the Echo Nest song ID overlaps with the MSD, which is a freely-available collection of audio features and metadata for a million contemporary popular music tracks. The MSD therefore enables researchers look into the songs by many acoustic features and metadata, it doesn't provide the audio file though.

Last.fm dataset 360K collects <*user*, *artist*, *plays*> triplets from Last.fm API. *plays* has two parts: *song title* and *play count*. The triplets don't have any time stamps, and they are not in chronological order neither. The dataset contains 359,347 unique users and 294,015 artists and 17,559,530 <*user*, *artist ID*, *plays*> triplets. Moreover, more information about users is available, including *gender*, *age*, *country* and *date* (The *date* description is not found). The song metadata and social tags are searchable on Last.fm by the song title and artist ID, but the acoustic features are not available on Last.fm.

Yahoo! Music User Ratings of Songs represents a snapshot of the Yahoo! Music community's preferences for various songs. It contains over 717 million

[4]http://labrosa.ee.columbia.edu/millionsong/

[5]http://blog.echonest.com/post/11992136676/taste-profiles-get-added-to-the-million-song-dataset

Dataset	Taste Profile Dataset	Last.fm dataset 360K	Yahoo! Music
Content	*<user ID, song ID, play count>*	*<user ID, artist, plays>*	*<user ID, song ID, rate>*
Number of songs	**384,546**	294,015 artists	~136,000
Number of users	1,019,318	359,347	**~1,800,000**
Number of triplets	48,373,586	17,559,530	**~717,000,000**
Rating type	play count	play count	**rate**
Other information	**Features & metadata provided by MSD**	Metadata provided by Last.fm	genre

Table 2: Available dataset comparison.

ratings of 136 thousand songs given by 1.8 million users of Yahoo! Music services. The rating triplet is *<user ID*, *song ID*, *rate>*, which *rate* is an integer from 1 to 5. All information of each song in the dataset is artist, album and genre. The users, songs, artists and albums are represented by randomly assigned numeric id so there is no identifying information revealed. Consequently, it is impossible to associate with other music dataset to obtain more information of songs.

The dataset comparison is summarized in Table 2. Although the Taste Profile Dataset doesn't provide explicit rating values, both features and metadata provided by MSD supply us for the probability of looking into acoustic content of a track, and play counts could implicitly represent rating scores.

Chapter 3

Feature Evaluation for Favorite Music Detection

In the recent digital world, millions of digital songs are available online and it is difficult for users to manually search favorite songs. A music recommender system is the solution that helps users find songs that they prefer from among this ocean of digital content, and it can lead the user along a personalized journey of song listening, song by song. The essential task of a recommender system is to be able to respond to the user's request: list songs that I prefer. Automatically detecting favorite songs is therefore an important part in a music recommender system. Basically, recommender systems apply for content-based methods, Collaborative Filtering (CF) techniques, or both to detect favorite songs.

Audio content-based methods use favorite songs to predict other songs users may like [Hoashi et al., 2003], based on their similarity to the favorite songs. In order to recommend favorite songs to users, a music recommender system using content-based analysis depends on manual or automatic audio content description to compute the similarity among songs.

For instance, Pandora[1] employs a team of musician analysts to listen to music and give each recording its descriptions, which includes melody, harmony, instrumentation, rhythm, vocals, lyrics and etc. A weighted Euclidean distance is used to find similar songs [Celma, 2010]. Z. Cataltepe et al. present a music recommender system based on audio similarity and users' listening histo-

[1]http://www.pandora.com/about/mgp

ry [Cataltepe and Altinel, 2007]. An industrial-strength music recommender system introduces the Euclidean distance of two tracks over a reduced space using Principal Component Analysis [Cano et al., 2005]. The system uses several audio features, including Mel-Frequency Cepstral Coefficients (MFCC), tempo, key mode and others.

Music was one of the first forms of content to be addressed by collaborative filtering recommender systems such as Ringo, Firey and HOMR [Shardanand, 1994] [Shardanand and Maes, 1995]. This technique finds users with similar music preferences and recommends items liked by these similar users to the target user [Konstas et al., 2009]. It could be seen as a method to measure the similarity based on the user preference. Hence, the user preference is considered as one type of feature in our work. Some hybrid methods combined with collaborative filtering techniques and content-based methods are proposed to solve the cold-start problem [Yoshii et al., 2006] [Donaldson, 2007a]. B. McFee et al. presented a method for deriving item similarity from a sample of collaborative filter data, and use the sample similarity to train a distance metric over acoustic features. The trained distance metric is used to improve the shortcoming of CF techniques, namely, the incompleteness of the data hampers the training.

This chapter focuses on systemic analysis regarding how important both audio content features and collaborating filtering signals are for favorite songs detection in the real world [Hu et al., 2013]. The two evaluation methods are described in Section 3.1. In Section 3.2, we look into the music log, and present the evaluation results. We make the conclusions and discuss the future work in Section 3.3.

3.1 Evaluation methods

We evaluate the importance of a type of feature (In the following paragraphs, “a feature” means “a kind of feature” instead of a value in a feature.) by the correlation between the predicted scores using this feature and the scores by real play count. Moreover, the favorite song detection problem could be converted to a binary classification problem. As a result, the feature selection results are used to evaluate the feature importance.

3.1.1 Correlation evaluation

Some content-based recommender systems assume that a song with high similarity to the favorite songs would be liked by the user [Cano et al., 2005] [Cataltepe and Altinel, 2007]. Thus, the systems measure the similarity among songs, and recommend the song with the highest similarity to the favorite songs. Euclidean distance and Dynamic Time Warping (DTW) algorithm are used to measure the similarity between two songs.

Euclidean distance sees a vector as a point in Euclidean space, and the distance between points is given by the Pythagorean formula. This metric works non-sequential features, like tempo.

For sequential feature, like pitch, DTW algorithm is a good metric to measure the distance between two sequences, which may vary in time, since DTW algorithm is able to find the optimal alignment between two time series [Salvador and Chan, 2007].

One time series may be non-linearly “warped” to the other one by stretching or shrinking it along its time series.

If a certain feature is important, the song similarity according to the feature should be highly effective to detect whether a song is a user favorite. The highly effective detection is expected to give high relevant rating score, and the predicted rating score should correlate or anti-correlate the actual one. Thus, correlation between the predicted scores and the actual scores, which is estimated by play count, is therefore used to evaluate the importance of the feature. The correlation result is normalized to $[-1.0, 1.0]$, which -1.0 is in the case of a perfect negative (decreasing) linear relationship, and +1.0 means a perfect positive (increasing) linear relationship. The greater the absolute value is, more important the feature is.

In a user’s log, the predicted rating score is given by the following equation.

$$\hat{r}_i^u = \frac{\sum_{s_j \in \mathbf{R}^u, j \neq i} r_j^u \cdot sim_f\left(s_i, s_j\right)}{\sum_{s_j \in \mathbf{R}^u, j \neq i} sim_f\left(s_i, s_j\right)}, \tag{1}$$

where $\hat{r}_i^u$ denotes the predicted score for song s_i, and $\mathbf{R}^u$ is the set of rated songs. Rating score r_j^u is estimated by the play count of song s_j. $sim_f\left(s_i, s_j\right)$ is the similarity between song s_i and song s_j in terms of feature f.

We apply this approach to predict the rating score for each song listened by user u, and get the predicted scores $\{\hat{r}_1^u, \hat{r}_2^u, ..., \hat{r}_n^u\}$. The correlation between $\{\hat{r}_1^u, \hat{r}_2^u, ..., \hat{r}_n^u\}$ and $\{r_1^u, r_2^u, ..., r_n^u\}$ are used to evaluate the importance of feature f.

Furthermore, CF technique predicts the interest of a user in a song by collecting preferences or taste information from other users. This technique is based on the assumption that a user A is more likely to have a user B's opinion on a song x than to have the opinion on x of a user chosen randomly, if A has the same opinion as B on many songs. A user's preference or taste is represented by the opinion on the songs the user listened.

We consider a user as a document, and treat the songs which the user listened as terms in the document. Then, the user could be represented by a vector of songs using TF-IDF values. The cosine distance between two vectors is used to measure the similarity between the two corresponding users.

Then, we measure the similarity between two songs considering user similarity and rating scores by the following equation.

$$\begin{aligned}
&\text{if } \ |\mathbf{U}_i| \le |\mathbf{U}_j| \\
&\quad sim\left(s_i, s_j\right) = \\
&\frac{\sum_{u_m \in \mathbf{U}_i} \left| \lambda \max\limits_{u_n \in \mathbf{U}_j} sim(u_m, u_n)^2 - \eta \left[\left(r_i^m - r_j^n \right) / V \right]^2 \right|}{|\mathbf{U}_i|}, \\
&\text{otherwise} \\
&\quad sim\left(s_i, s_j\right) = sim\left(s_j, s_i\right),
\end{aligned} \tag{2}$$

where $\mathbf{U}_i$ is the set of users who played song s_i, and u_m denotes the user m. V is the rating scale. λ and η are two regulatory factors.

This measurement allows the similarity of two songs to be high, if two users have high similar tastes, and rate the two songs at the same score. On the contrary, if two users have high similar tastes, and rate two songs at totally different scores, or the two users' tastes are different but the ratings are similar, the two songs must be different in terms of user preference.

This type of similarity is also used to predict song ratings by Equation 1, in order to evaluate the importance of user preference.

3.1.2 Feature selection methods evaluation

The classification of a song according to "like" or "dislike" is a binary classification problem. We compared several feature selection methods, and selected χ^2 Statistic (Chi), Information Gain (IG), Information Gain Ratio (IG Ratio) and Uncertainty to evaluate the importance of features, respectively.

χ^2 Statistic is used to investigate how much a feature depends on the category by the following equation.

$$\chi^2 = \frac{(ad - bc)^2 (a + b + c + d)}{(a + b)(c + d)(b + d)(a + c)}, \tag{3}$$

where a, b, c and d are the number of instances in different cases as shown in Table 3. If the feature is categorical, "Data Type" is the category. If the feature is numerical, "Data Type" is the data range.

IG evaluates the importance of a feature from the view of information theory. Essentially, the expected IG is the change in information entropy from a prior state to a posterior state that takes some information as given:

Feature	Data Type 1	Data Type 2	Total
Category 1	a	b	a+b
Category 2	c	d	c+d
Total	a+c	b+d	a+b+c+d=N

Table 3: General notation for a 2 x 2 contingency table.

$$\mathrm{IG}\left(T,f\right)=\mathrm{H}\left(T\right)-\mathrm{H}\left(T|f\right), \tag{4}$$

where $\mathrm{IG}\left(T,f\right)$ denotes the IG of the feature f on the training data T. $\mathrm{H}\left(T\right)$ is the entropy of the training data, and $\mathrm{H}\left(T|f\right)$ is the average conditional entropy of T given f.

IG Ratio introduces Intrinsic Value (IV) to evaluate the feature importance, and it is the ratio between the information gain and the intrinsic value.

$$\mathrm{IGR}\left(T,f\right)=\mathrm{IG}\left(T,f\right)/\mathrm{IV}\left(T,f\right), \tag{5}$$

where $\mathrm{IGR}\left(T,f\right)$ is the IG Ratio of feature f and $\mathrm{IV}\left(T,f\right)$ is the intrinsic value given by:

$$\begin{aligned}\mathrm{IV}\left(T,f\right)=&\\ &-\sum_{v}\frac{|\{t\in T|t_f=v\}|}{|T|}\log\left(\frac{|\{t\in T|t_f=v\}|}{|T|}\right)\end{aligned} \tag{6}$$

Uncertainty measures the relevance of a feature by calculating the symmetrical uncertainty with respect to the class.

$$\mathrm{Uncentainty}\left(f\right)=2\cdot\frac{\left(p\left(c\right)-p\left(c|f\right)\right)}{p\left(c\right)+p\left(f\right)} \tag{7}$$

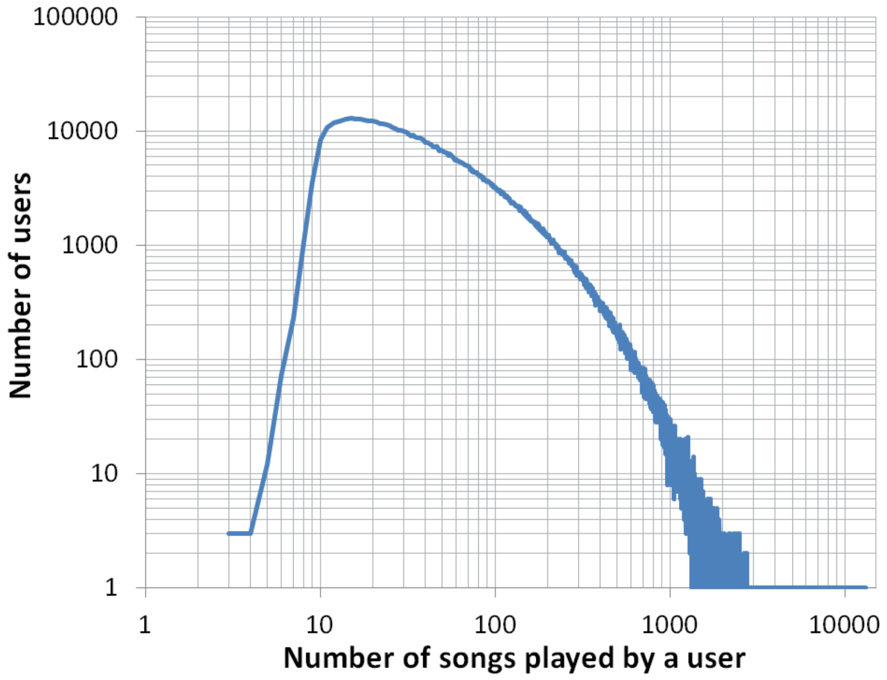


Figure 6: Distribution of users by how many songs are played.

These four methods evaluate the feature importance from different views, and Chi-square statistic and Information Gain have been proved that they surpass among feature selection methods in a text classification task [Forman, 2003]. We therefore apply each of them to evaluate the feature importance.

3.2 Analysis results

An ideal analysis is expected to run on a dataset which contains a great number of songs and users. Furthermore, the users should explicitly rank every song in the dataset. However, it is impractical. What's worse, regarding to the private information protection, the exposed information is limited.

Compared to other available dataset, aforementioned in Section 2.3, we use Taste Profile Dataset to do experiments. A closer look at the Taste Profile Dataset presents the distribution of users by the number of songs, as shown in Figure 6.

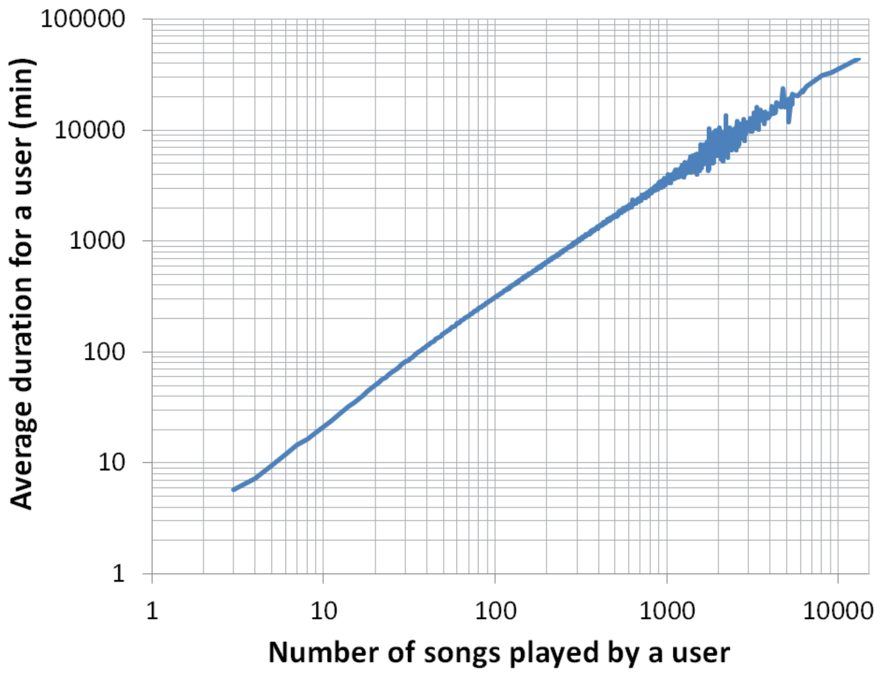


Figure 7: Distribution of estimated duration.

The user logs, of which the number of played songs is less than three, are removed by the Taste Profile dataset. The distribution mainly locates in [10, 100], and it has a long tail.

A user's preference would be changed by different contexts in a long period so the overall listening duration of a user should be taken into account. The Taste Profile dataset doesn't show the actual listening duration of a song. We roughly estimate how long a user listens to the songs in the user log by Equation 8.

$$T = \sum_{i \in S}^{i} n_i \cdot t_i \cdot \left(1 - \frac{1}{n_i + 1}\right), \tag{8}$$

where n_i is the play count of song i by the user. t_i is the duration of song i, which can be obtained from MSD. The estimated duration of playing songs by users is shown in Figure 7.

Category	Features
Acoustic features	bars, beats, sections, segment, **loudness**, **pitches**, timbre, tatums, key, mode and **tempo**
Song metadata	sample rate, **duration**, release, **song hotness**, title and year
Artist metadata	terms, **similar artists**, **artist hotness**, **artist familiarity**, artist location, artist name and musicbrainz tags

* Bold font means that the feature is selected.

Table 4: Features provided by MSD.

3.2.1 Evaluation settings

MSD provides many types of features for a song as shown in Table 4. We select eight features from the MSD, i.e., *loudness*, *pitches*, *tempo*, *duration*, *song hotness*, *artist similarity*, *artist hotness* and *artist familiarity*, and these features cover the three categories in Table 4. Furthermore, user preference is counted in too. The similarity is able to indirectly represent the performance of CF techniques in favorite song detection.

Euclidean distance is applied to measure the distance of songs for non-sequential features, while DTW method measures the distance of songs for sequential features, like pitches. The quadratic time and space complexity of DTW limits its use to only small time series data. Thus, we apply FastDTW [Salvador and Chan, 2007] to accelerate the similarity measurement to O(n) time.

Play count is considered as an implicit rating based on the assumption that the rating is positive if the song is played many times, and vice versa. Considering the bias of the ratings, the normalized rating $r'_{i,j}$ from -1.0 to 1.0 is given by:

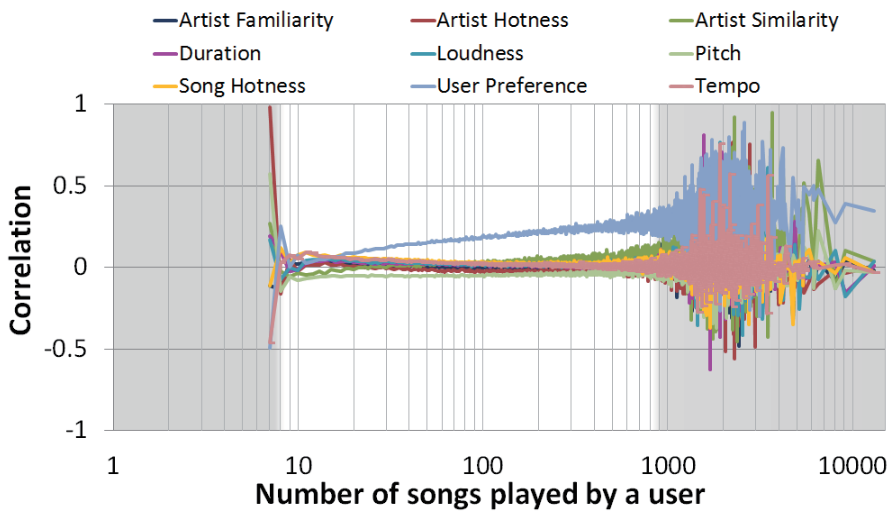
$$r_i^{j\prime} = r_i^j - \frac{1}{2}\left(\bar{r}_i^{\cdot} + \bar{r}_{\cdot}^j\right), \tag{9}$$

where $\bar{r}_i^{\cdot}$ is the average rating of all ratings to song s_i, and $\bar{r}_{\cdot}^j$ denotes the average rating by user u_j.

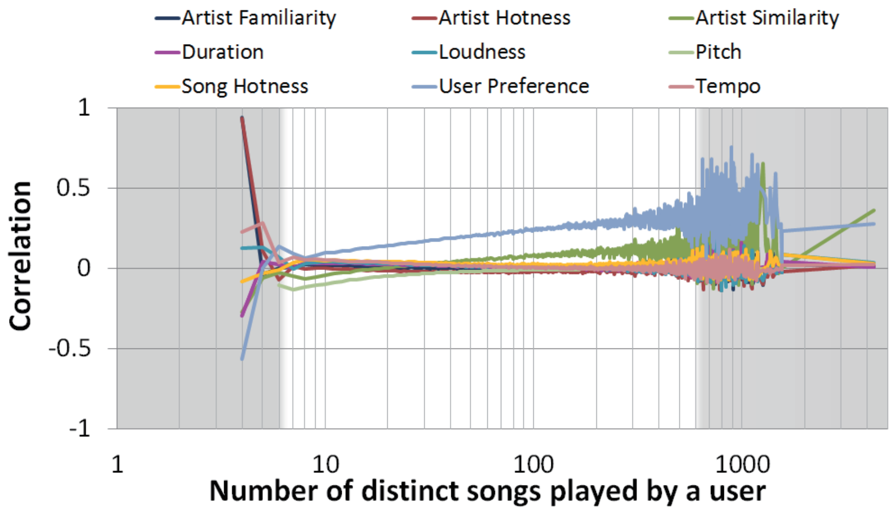
We evaluate the feature importance by the correlation between predicted scores and normalized actual scores as described in Section 3.1.1. As all non-sequential features but user preference could represent a song to a vector, they are evaluated by feature selection methods mentioned in Section 3.1.2. The numeric value of a feature is assigned into categories by predefined window to apply feature selection methods. The normalized rating less than -0.1 is seen as "dislike" while the song is beloved if the rating greater than 0.1. Because the ratings between -0.1 and 0.1 is blur to classify the song to like or dislike, these ratings are ignored. The evaluation result is normalized to [0.0, 1.0] in order to compare the results among different selection methods.

3.2.2 Results and discussion

The value of each plot in Figures 8 and 9 is the mean of the results by different users who play the same number of songs. Figure 8 shows the correlation between the predicted ratings and the normalized ratings by play count in two views, including nine features. Figure 8(a) shows the distribution of the correlation by the overall played songs. In Figure 8(b), the correlation varies by the unique songs, which means that the available information about songs increases as the unique songs increases.



(a) Based on overall songs



(b) Based on unique songs

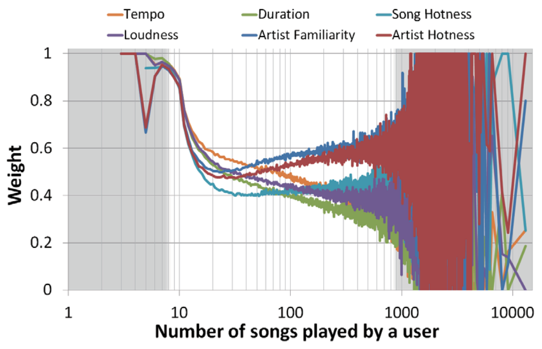
Figure 8: Correlation between predicted and actual ratings.

In Figure 8, the gray shadow covers the number of played songs by which there are fewer than 30 users so that the correlation results in the shadow are not statistically reliable. In the remaining part, user preference similarity is the most important feature except at the cold start, namely, CF signal is remarkable for favorite song detection. Basically, artist similarity is the secondary important feature. The other curves fluctuate around 0.0, which means the corresponding features are not critical for favorite song detection.

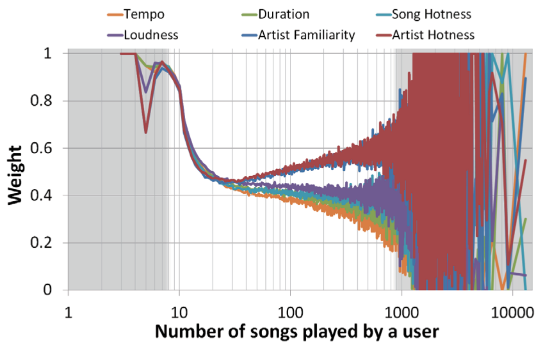
At the cold start period, it is frustrating that no selected features makes a great contribution to favorite song detection. As the number of played songs increases, the number of users reduces, and the fluctuation of curves becomes vast but the main trend doesn't change significantly. Thus, in a long playing period, the importance of user preference is basically stable.

We leverage feature selection methods to measure the feature importance, and the feature selection results are normalized to [0.0, 1.0]. Zero means the feature doesn't play an important role and one means the feature is crucial. Figure 9 presents the evaluation results. The gray shadow also covers the region that the number of users is less than 30.

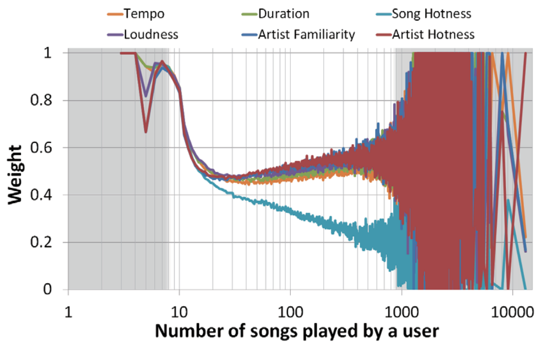
When the number of played songs is less than 10, most features are highly effective to distinguish "like" and "dislike". Then, the curves remarkably separate. As the play counts increase, artist hotness and artist familiarity rise and other features descend more or less except Figure 9(c). Referring to Figure 7, the importance of features varies by the overall duration. In a long period, the user



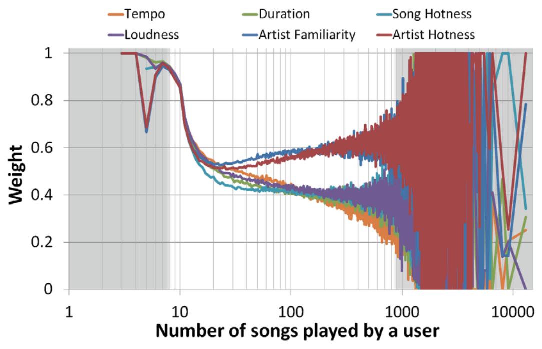
(a) Chi Squared Statistic



(b) Information Gain



(c) Information Gain Ratio



(d) Uncertainty

Figure 9: Feature selection results by different feature selection methods.

preferences would be changed by different contexts. As a result, the importance of tempo, duration, song hotness and loudness becomes weak while that of artist familiarity and artist hotness grows. Therefore, artist familiarity and artist hotness are more stable in a long listening period than other features.

3.3 Conclusions and future work

In this chapter, we compare nine features by correlation and feature selection methods. Among these features, user preference and artist similarity play important roles in favorite song detection. Artist familiarity and artist hotness are stable in a long listening period. The evaluation results show that collaborative filtering technique has high performance for favorite song detection. Song hotness is not as important as people thought.

If audio files of songs are available, we would employ a feature extraction tool to gain more features, and compare them in the future. If more information on the user log is exposed, such as time stamps and sequence information, the preference variation model could be analyzed.

Chapter 4

Music Recommendation for Individual Users

Even though favorite songs could be detected accurately, directly recommending favorite songs is not good enough for users. For example, some online music websites supply top n songs ranked by current popularity. These songs are very much likely to be loved by users. However, if a recommender plays these songs one by one, it might make users disappointed. A good music recommender is therefore supposed to serve favorite and novel songs according to the current context. In this chapter, we focus on recommending music on a local device, since the purchased or downloaded tracks are mostly loved by the user currently or ever.

A user's feedbacks are precious treasure for a recommender to understand the user's preference and improve the recommendation quality. However, listening to music is to relax but not to do a survey for identifying user preferences on music. As an extra cost for listening music, the effort required to provide feedbacks should be minimized. Meanwhile, the feedback should maximize the user's satisfaction by playing appropriate songs at the right time. We evaluate the user's attitude towards a song by the partition of playing time. In particular, if a song is played from the start to the end, we infer that the user likes the song and the recommendation is satisfying. On the other hand, if the song is skipped in a few seconds, we assume that the user dislikes the song at that time, and the recommendation is less effective.

4.1 Recommendation approach

We predict genre and publish year of the next song, because some users like listening to similar songs according to genre and publish year, whereas others maybe love mixing songs on genre and publish year. Hence, the intuitive assumption that the song similar to the current one is a good choice for recommendation is not as good as it is. We apply time series analysis to predict genre and publish year of the next song.

Obviously, the system should recommend the user's favorite songs to them. How many times a song has been actively played and how many times the song has been completely played can be used to infer the strength of favor of the song. We collected a user's behavior to analyze the favor of songs.

In common sense, a few users like listening to a song many times in a short period, even though the song is loved by the user. On the other hand, some songs that the user favored many months ago may be old and insipid currently. However, if the system recommends them at right time, the user may feel that they are fresh. Consequently, we take the freshness of songs into account.

Due to activities and biological clock, users have different tastes in choosing music at different time. In a different period of a day or a week, users tend to select different songs. For example, in the afternoon, a user may like soft music for relaxation while the use love energetic songs in the evening. In this chapter, we use a Gaussian Mixture Model to represent the time pattern of listening, and predict the probability of playing a song at that time.

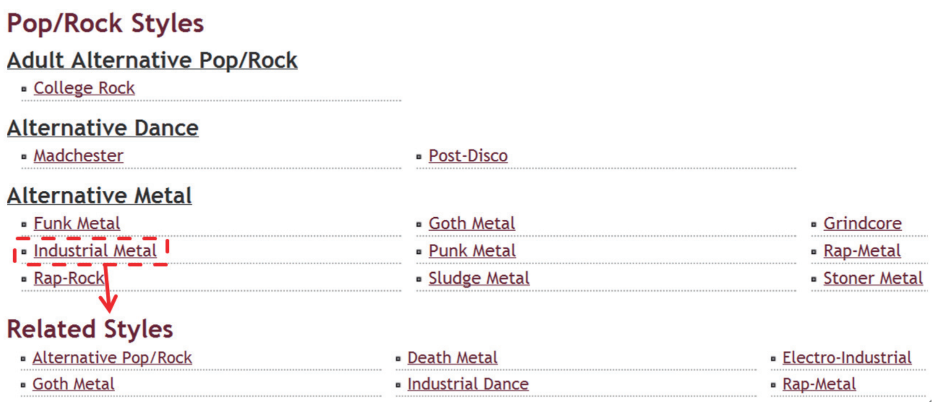
All in all, determining which song is to be recommended as the next one in the playlist is based on five perspectives: genre, year, favor, freshness and time pattern. The weights for these factors vary based on the user's feedbacks [Hu and Ogihara, 2011].

4.1.1 Genre

The sequence of recent playing of a user represents the user's habit of listening, so we analyze the playing sequence using a time series analysis method to predict the genre of the next song. The system records the recent 16 songs that were played at least a half of their length. Although most of the songs record their genres and years in ID3v1 or ID3v2 tags, some of tags are notoriously noisy. Hence, we developed a web wrapper to collect genre information from AllMusic.com, a popular music information website, and use that information to complement songs' genres. When AllMusic.com has no information about a track, we leverage a trained genre classifier to predict the genre label. The genre classifier will be introduced in Section 4.2.

Furthermore, AllMusic.com not only has a hierarchical taxonomy on genre but also provides related genres of subgenres. The hierarchical taxonomy and related genres are shown in Figure 10. For example, `Industrial Metal`, whose parent is `Alternative Metal`, is related to `Alternative Pop/Rock`.

We use the taxonomy to build an undirected graph, in which each node describes a genre and each edge's value is the distance between two genres. The values of the graph are initialized by a maximum value. The parent and related



Pop/Rock Styles

Adult Alternative Pop/Rock
- College Rock

Alternative Dance
- Madchester
- Post-Disco

Alternative Metal
- Funk Metal
- Goth Metal
- Grindcore
- Industrial Metal
- Punk Metal
- Rap-Metal
- Rap-Rock
- Sludge Metal
- Stoner Metal

Related Styles
- Alternative Pop/Rock
- Death Metal
- Electro-Industrial
- Goth Metal
- Industrial Dance
- Rap-Metal

Figure 10: Genre taxonomy screenshot in AllMusic.com.

relationship are valued at different distances, which varies by the depth in the taxonomy, that is, high level corresponds to large distance while low level corresponds to small distance. Then, we assume the distance measurement is transitive and update the graph until no edge changes.

$$E_{ij} = \min_k \left(E_{ij}, E_{ik} + E_{kj}\right), \tag{10}$$

where E_{ij} is the value of edge (i, j). Therefore, we obtain the distance between any two kinds of genre and the maximum value in the graph is 6.

Now, the system converts the series of genres of recent songs into a sequence of distance between neighbor genres using the distance matrix. The sequence of distance will be seen as the input for time series analysis method, and we can estimate the next distance. Then, the current genre and the estimated distance will give us genre candidates.

Autoregressive Integrated Moving Average (ARIMA) [Box and Pierce, 1970] model is a general class of models in time series analysis. An ARIMA(p, d, q) model can be expressed by following polynomial factorization.

$$\Phi\left(B\right)\left(1-B\right)^{d}y_{t}=\delta+\Theta\left(B\right)\varepsilon_{t}, \tag{11}$$

$$\Phi\left(B\right)=1-\sum_{i=1}^{p}\phi_{i}B^{i}, \tag{12}$$

$$\Theta\left(B\right)=1+\sum_{i=1}^{q}\theta_{i}B^{i}, \tag{13}$$

where y_t is the tth value in the time series of data $\mathbf{Y}$, and B is the lag operator; ϕ and θ are the parameters of the model, which are calculated in analysis; p and q are orders of autoregressive process and moving average process, respectively; And d is a unitary root of multiplicity.

The first step of building ARIMA model is model identification, namely, estimating p, d and q by analyzing observations in time series. Model identification is beneficial to fit the different pattern of time series. The second step is to estimate parameters of the model. Then, the model can be applied to forecast the value at $t+\tau$, for $\tau>0$. As an illustration consider forecasting the ARIMA$(1,1,1)$ process

$$\left(1-\phi B\right)\left(1-B\right)y_{t+\tau}=\left(1-\theta B\right)\varepsilon_{t+\tau}, \tag{14}$$

$$\hat{\varepsilon}_{t}=y_{t}-\left[\delta+\sum_{i=1}^{p+d}\phi_{i}y_{t-i}-\sum_{i=1}^{q}\theta_{i}\hat{\varepsilon}_{t-i}\right]. \tag{15}$$

Our system uses ARIMA to fit the series of distance and to predict the next distance. The process is shown in Figure 11.

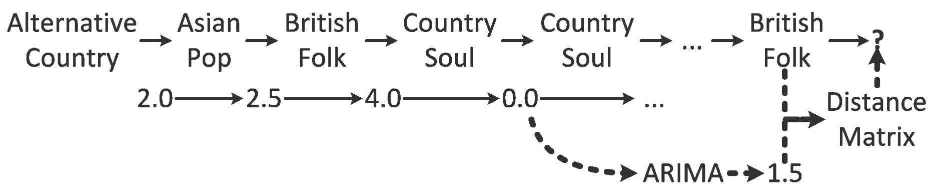


Figure 11: Predict the next genre.

We use Gaussian distributions to evaluate each possible genre for the next track, and select the one with the biggest probability.

4.1.2 Publish year

The publish year is similar to genre so we use ARIMA to predict the next possible year and compute the probability of the next recording year.

4.1.3 Freshness

As a new contribution of this approach, we take freshness of a song for a user into consideration. Many recommendation systems [Logan, 2004] based on metadata of music and user behavior cannot avoid recommending the same music under the same situations. What's worse is that these songs will still be at the top of recommendation result, since they have been recommended and played many times, and are seen as favorite songs. Because the rich get richer, a small set of songs will be recommended again and again. The iterations make users fall into a "favorite trap" and feel bored. Therefore, an intelligent recommendation system should avoid recommending the same set of songs many times in a short period. On the other hand, the system is supposed to recommend some songs that which not been played for a long time, because these songs are fresh to users even though they once listened to them multiple times.

Freshness can be considered as the strength of strangeness or the amount of experience forgotten. We apply Forgetting Curve [Ebbinghaus, 1913] to evaluate the freshness of a song to a user. Forgetting Curve is shown as follows.

$$R = e^{-\frac{t}{S}}, \tag{16}$$

where R is memory retention, S is the relative strength of memory and t is time.

The less memory of a song remains in a user's mind, the fresher the song is for the user. In our work, S is defined as the number of times the song has been played, and t is the distance of present time to the last time the song was played. The reciprocal of memory retention is normalized to represent the freshness.

This metric contributes towards selecting fresh songs as recommendation results rather than recommending a small set of songs repetitively.

4.1.4 Favor

The strength of favor for a song plays an important role in recommendation, because a recommendation system should give high priority to user's favorite songs. User behavior can be implied to estimate how favored the user feels about the song based on a simple assumption: A user listens to a favorite song more often than others, and often listens to a larger fraction of the favorite song than the other.

We consider the favor of a song from four counts: active play times, passive play times, skip times and delete times. Passive play time means the song is played as a recommendation result or as the next one in playlist. The favor is assessed by the weighted average of the four factors.

4.1.5 Time pattern

Since a user would have different habits or tastes in different periods of a day or a week, our recommendation system takes time pattern into account based on the user log. The system records the time of the day and week when songs are played. It then employs Gaussian Mixture Model to estimate the probability of playing a song at a specific time. The playing time of a song in different periods trains the model using Expectation Maximization algorithm. When the system recommends songs, the model is applied to estimate the probability of the song being played at that time.

4.1.6 Integration of scores

A song is assessed how good it is for recommending as the next song from the five perspectives described as the above. In order to rank results and make a selection, the scores should be integrated into a final score. At first, the scores are normalized into the same scale. Since different users have different tastes, these five factors are assigned different weights at integration. We are inspired by Gradient Descent to match users' need. However, it is not friendly for a user to show several possible recommendations to select, the interaction is though useful to determine where to descent. Consequently, we use the recent recommendation results to adjust the weights, which is initialized by $(1.0, 1.0, 1.0, 1.0, 1.0)$. The algorithm is shown in Algorithm 1. This algorithm adjusts the weights when users don't satisfy the current recommendation. The biggest weight is reduced by step δ, and other weights share the change, δ.

Algorithm 1: Adjust weights based on recent recommendation results.

```
Input: Recent $k$ recommendation results $\Re_t\left(R_{t-k+1}, R_{t-k+2}, \ldots, R_{t-1}, R_t\right)$
at time $t$.
$R_i$ contains user interaction of this recommendation $\chi_i$, which is like or
dislike, and the score of each factor of the recommendation $i$ is $\mathbf{\Lambda}_i$.
Descent step $\delta$, which is positive.
Current factor weights, $\mathbf{W}$.
Output: New factor weights, $\mathbf{W}'$.
Process:
if $\chi_t$ = dislike then
    Initialize an array $\mathbf{F}$ to record the contribution of each factor.
    for $i = R_{t-k+2}$ to $R_t$ do
        $\Delta\mathbf{\Lambda}_i = \mathbf{\Lambda}_i - \mathbf{\Lambda}_{i-1}$
        $max = \arg\max_j \left(\Delta\lambda_j\right), 1 \leq j \leq 5$
        $min = \arg\min_j \left(\Delta\lambda_j\right)$
        if $\chi_i = Like$ then
            $\mathbf{F}_{max} = \mathbf{F}_{max} + 1$
        end
        else
            $\mathbf{F}_{max} = \mathbf{F}_{max} - 2$
            $\mathbf{F}_{min} = \mathbf{F}_{min} + 1$
        end
    end
    $inIndex = \arg\max_j (\mathbf{F})$
    $w'_j = \begin{cases} w_j + \delta, j = inIndex \\ w_j - \delta/4, otherwise \end{cases}, j = 1, 2, 3, 4, 5$
    $deIndex = \arg\min_i (\mathbf{F})$
    $w'_j = \begin{cases} w_j - \delta, j = deIndex \\ w_j + \delta/4, otherwise \end{cases}, j = 1, 2, 3, 4, 5$
end
else
    $\mathbf{W}' = \mathbf{W}$
end
return $\mathbf{W}'$
```

4.1.7 Cold start

Cold start is a difficult problem to tackle for recommendation systems. When a recommendation system begins with no idea as to what kinds of songs users like or dislike, it hardly gives any valuable recommendation. As a result, in the cold start, the system randomly picks a song as the next song and records the user's interaction, which is similar to Pampalk's work [Pampalk et al., 2005]. After 16 songs have been played, the system uses the metadata of these songs and user behavior to recommend a song as the next one.

4.2 Genre classification

In music recommendation and music information retrieval, many methods see song genre as important metadata for retrieving songs. Although genre is sometimes stored in the header of MP3 file, like ID3v1 and ID3v2, some MP3 files miss genre labels in the headers or have a noise label. A reliable classifier is therefore expected to complement genre labels. As a musician attribute, genre is dependent on music content, so, obviously, acoustic features are a valuable data source to predict genre class. Furthermore, other data sources, like lyrics, social tags, and artist tags, also provide useful information for genre classification.

Some papers have discussed the importance of multiple data sources in genre classification and proposed methods to use them. Most of these methods [McKay et al., 2010] concatenated features from different data sources into a vector to represent the song. However, for a huge scale dataset, it is impractical to require that every instance must have valid data in all data sources. It is inevitable for the classification results to be demoted due to data missing in the concatenated vector.

If we imagine classifiers as experts in voting, the authority of an expert could be assessed by the accuracy of the corresponding classifier. Because both the types of input data and learning methods are different, the views of experts are not same. Therefore, the confidences to make a correct decision regarding a particular item are also different. Hence, the voting result of an instance is related to both the authority of the classifier and the confidence of the classifier to classify the particular instance [Hu and Ogihara, 2012].

We extract features from audio, artist terms, lyrics and social tags to represent songs, and train sub-classifiers by each data source using the best classification method for the data source. These trained sub-classifiers are combined to classify song genre using the aforementioned intuitive idea. If a song misses data in certain data type, the corresponding classifier has zero confidence to classify the song. Hence, the song would be classified by the remained data without any negative influence caused by the missing data.

4.2.1 Combination method

We apply different data source to train sub-classifiers, respectively. We assume that classifier set C contains n sub-classifiers, namely, $C = \{c_1, c_2, \ldots, c_n\}$. Furthermore, we assume that songs are distributed into m genres, $G = \{g_1, g_2, \ldots, g_m\}$. The trained sub-classifiers use predicted labels to vote as shown in Equation 17.

$$\mathrm{G}\left(I_k\right) = \arg\max_{g_j} \left\{ \sum_{i=1}^{|C|} \left[\mathrm{Auth}\left(c_i\right) \cdot \mathrm{Conf}\left(c_i, g_j, I_k\right)\right] \right\} \tag{17}$$

$\mathrm{Auth}\left(c_i\right)$ denotes the authority of the classifier c_i from 0.0 to 1.0, and it is estimated by the accuracy of the classification in the validation test.

$\mathrm{Conf}\left(c_i, g_j, I_k\right)$ is the confidence of the classifier c_i to classify the instance I_k

Table 5: Data sources used to predict genre labels.

Name	Extracted information	Number of records
MSD	Audio features, artist terms	1,000,000
MuisXmatch	Lyrics features	237,662
Last.fm tags	Social tags	505,216

to genre g_j. For Naïve Bayes, the posterior probability is seen as the confidence for a class. Neural Net has normalized real value output from -1.0 to 1.0. The absolute value means the confidence to assign the instance to a positive or negative label. We employ the method proposed by Lee [Lee, 2010] to estimate the confidence for logistic regression. The margin from the instance location to the classification hyper plane is considered to be the confidence of the SVM classifier. The confidences of classifiers are normalized into $[0.0, 1.0]$. The confidence for invalid data is set to 0.0, in order to avoid negative effect caused by invalid data.

4.2.2 Classification evaluation

In our experiment, we collected features from MSD, MusiXmatch and Last.fm tag data sets, as shown in Table 5. The records in these data sources are matched via `trackID`.

AllMusic.com provides the genre taxonomy which consists of 10 major genres with sample songs. We collect 1,138 songs which has a valid record in MSD from AllMusic.com as the ground truth. The distribution of the songs according to genre is shown in Figure 12.

In order to improve classification performance, we convert genre classification into a series of binary classifications. Thus, the classification result of a song

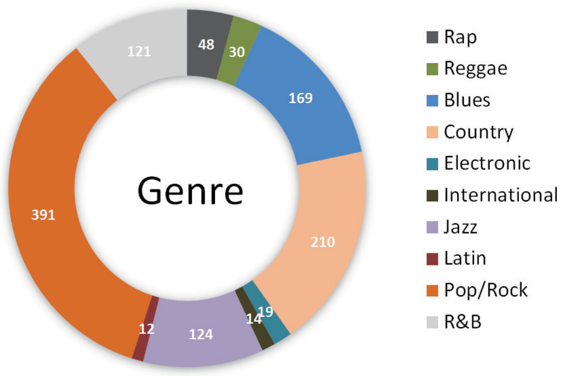


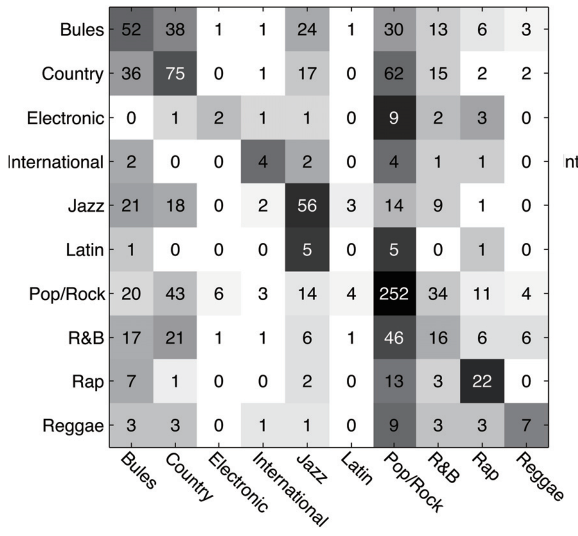
Figure 12: Distribution of sample songs with genre labels.

is a vector of probability to classify the song into a genre. The predicted genre is labeled as the one with the highest probability.

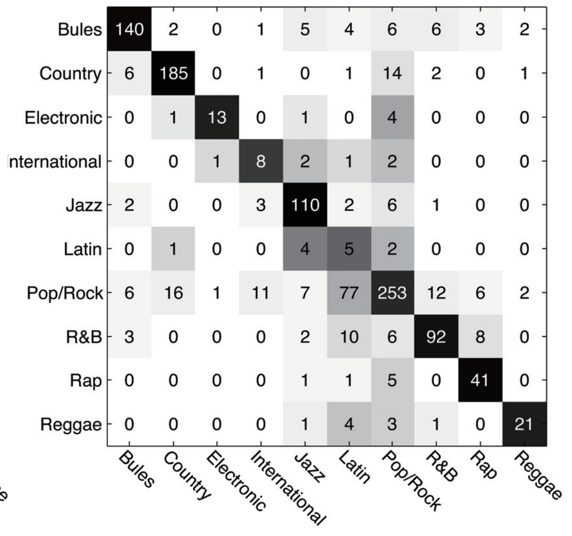
We extract features from different data sources, and trained individual classifiers by each type of features using Naïve Bayes, Rule Induction, LDA, Neural Net, Logistic Regression and SVM, respectively. The classifiers' performances are evaluated by 5-folder cross validation. The performance of the best classifiers in the types of features are shown in Figure 13.

The four sub-classifiers are combined based on classification authority and confidence. The resultant combined classifier is significantly better than its sub-classifiers, and it is further better than the SVM classifier trained by the concatenated vector from four data sources, as shown in Figure 14. The classification accuracy of different classifiers are summarized in Table 6. Furthermore, the result is encouraging regarding to the result of genre classification task in MIREX[1]. We apply the combined classifier to classify all of the songs in the MSD and the result is available at `http://web.cs.miami.edu/home/yajiehu/resource/genre`.

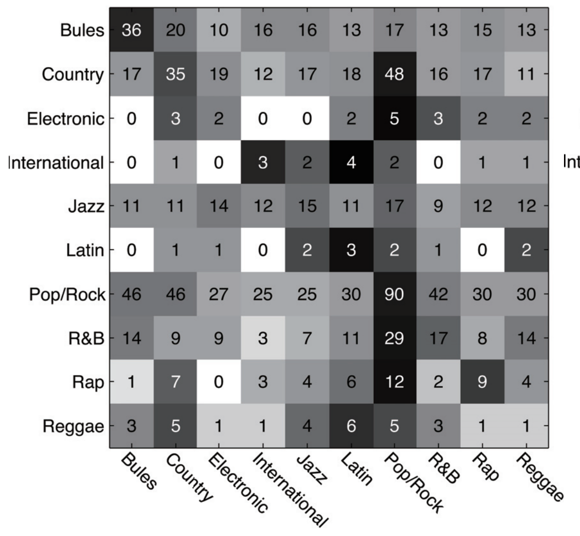
[1]http://www.music-ir.org/mirex/wiki/2009



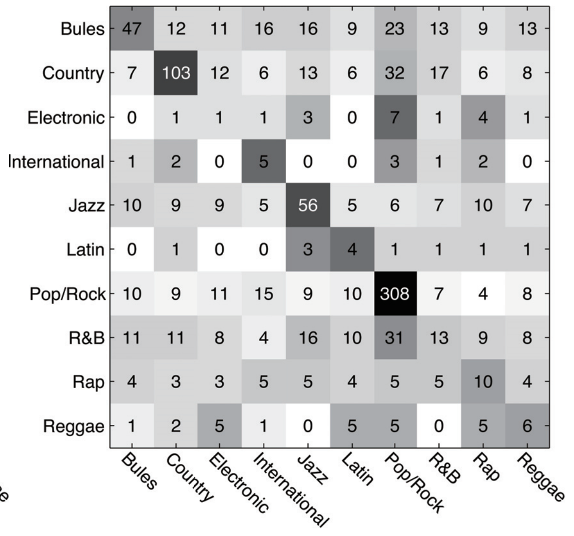
(a) Neural Net by audio features



(b) Neural Net by artist terms



(c) Logistic Regression by lyrics

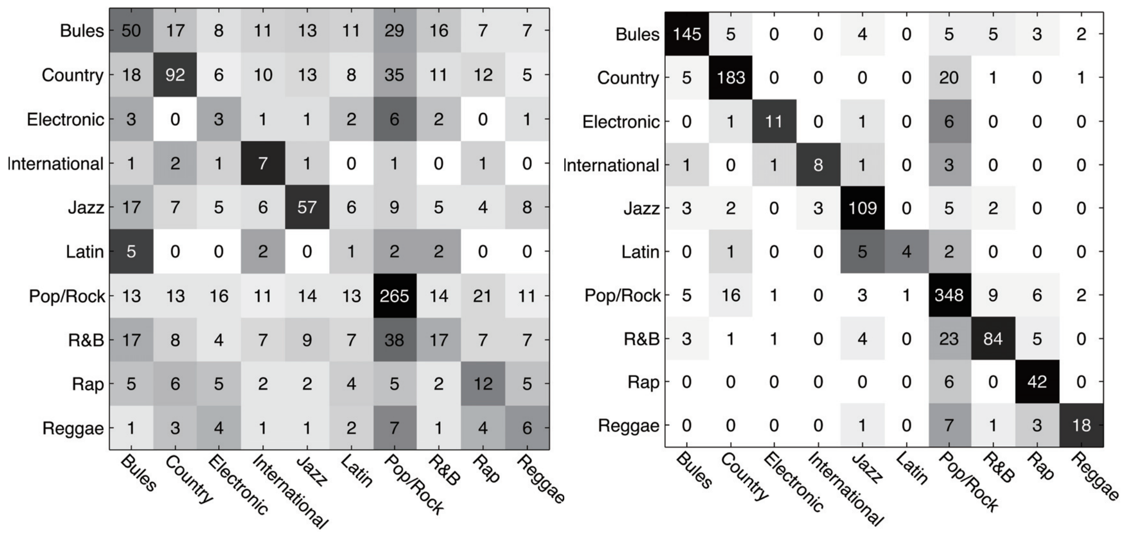


(d) Naïve Bayes by social tags

Figure 13: Confusion matrixes of four sub-classifiers.

Table 6: Experiment result comparison.

Data	Method	Accuracy
Audio	Neural Net	42.70%
Artist terms	Neural Net	76.27%
Lyrics	Logistic Regression	18.54%
Social tags	Naïve Bayes	48.59%
All data	SVM	44.82%
All data	Combined classifier	**83.66%**



	Bules	Country	Electronic	International	Jazz	Latin	Pop/Rock	R&B	Rap	Reggae
Bules	50	17	8	11	13	11	29	16	7	7
Country	18	92	6	10	13	8	35	11	12	5
Electronic	3	0	3	1	1	2	6	2	0	1
International	1	2	1	7	1	0	1	0	1	0
Jazz	17	7	5	6	57	6	9	5	4	8
Latin	5	0	0	2	0	1	2	2	0	0
Pop/Rock	13	13	16	11	14	13	265	14	21	11
R&B	17	8	4	7	9	7	38	17	7	7
Rap	5	6	5	2	2	4	5	2	12	5
Reggae	1	3	4	1	1	2	7	1	4	6

(a) SVM by long vector

	Bules	Country	Electronic	International	Jazz	Latin	Pop/Rock	R&B	Rap	Reggae
Bules	145	5	0	0	4	0	5	5	3	2
Country	5	183	0	0	0	0	20	1	0	1
Electronic	0	1	11	0	1	0	6	0	0	0
International	1	0	1	8	1	0	3	0	0	0
Jazz	3	2	0	3	109	0	5	2	0	0
Latin	0	1	0	0	5	4	2	0	0	0
Pop/Rock	5	16	1	0	3	1	348	9	6	2
R&B	3	1	1	0	4	0	23	84	5	0
Rap	0	0	0	0	0	0	6	0	42	0
Reggae	0	0	0	0	1	0	7	1	3	18

(b) Combined classifier

Figure 14: Confusion matrixes by all data.

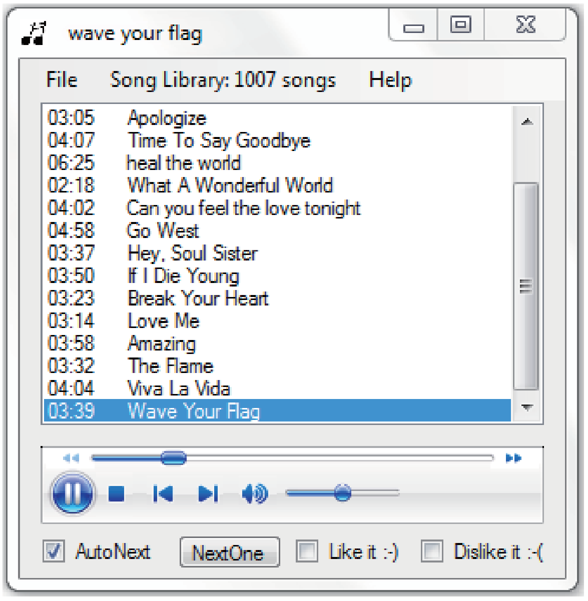
4.3 Recommendation experiment

The goal of recommendation systems is to cater to users' taste and recommend the next song at the right time and in the right order. Therefore, we focus on the real user experience to compare the user's satisfaction between our method and the baseline method, which randomly picks a song as the next one. We deem that most of the songs in a user's device are their favorite, but it doesn't mean that every song is proper to be played anytime. The feedback to random selections represents the quality of songs in users' devices, and the comparison result between our method and random selection shows the value of our method.

4.3.1 Data collection

An application system, named NextOne Player[2], is implemented to collect run-time data and user behavior for this experiment. The appearance of the

[2]Available at http://sourceforge.net/projects/nextoneplayer/



03:05	Apologize
04:07	Time To Say Goodbye
06:25	heal the world
02:18	What A Wonderful World
04:02	Can you feel the love tonight
04:58	Go West
03:37	Hey, Soul Sister
03:50	If I Die Young
03:23	Break Your Heart
03:14	Love Me
03:58	Amazing
03:32	The Flame
04:04	Viva La Vida
03:39	Wave Your Flag

Figure 15: The appearance of NextOne Player.

application is shown as Figure 15. It is developed in .NET Framework 4.0 using Windows Media Player Component 1.0. In addition to the functions of Windows Media Player, NextOne Player offers recommendation function using the proposed recommendation approach, and collects the user's feedbacks for performance evaluation. The recommendation will work when the current song in the playlist ends or `NextOne` button is clicked. The `Like it` and `Dislike it` buttons are used to collect the user's explicit feedbacks. The proportion of a song played is recorded, and it is considered as implicit feedbacks of the user for the song.

In order to compare our method to random selection, NextOne Player selects one of the two methods when it is launched. The probability of running each method is set to 0.5. Nothing but the recommendation approach is different, so, in the contrasting experiment, the user cannot realize which method is running.

We have collected data from 11 volunteers. They consist of 9 graduate students and 2 professors, including 3 female users. They use the application on their devices, and the recommended songs are from their own collections, so the experiment is run on open datasets.

4.3.2 Comparison results

First, we show the running time of the recommendation process, since it is a major influence on the user experience. We run the recommendation system for different magnitudes of the song library, and the system recommends 32 times at each size on a laptop[3]. Figure 20 shows the variation in running time by the size of song library. We observe that the running time increases linearly with the increase of the library size, and the running time is in an acceptable range. In order to provide a user-friendly experience, the recommendation is processed when the playing position is approaching to the end of the current song, and the result is generated as soon as the current song ends.

In order to evaluate the approach, the system records the user's behavior. We collected the user logs from the volunteers, and calculated the average played proportion of the songs, which means how much partition of a song is played before it is skipped. Under the assumption that the partition implies the satisfaction of the user at the recommendations, we evaluate the recommendation approach by the partition as shown in Figure 17. The histograms in the chart represent the number of songs that were played on a day, whereas the curves represent the variation of the

[3]CPU: Intel i7, RAM: 4GB, OS: Windows 7

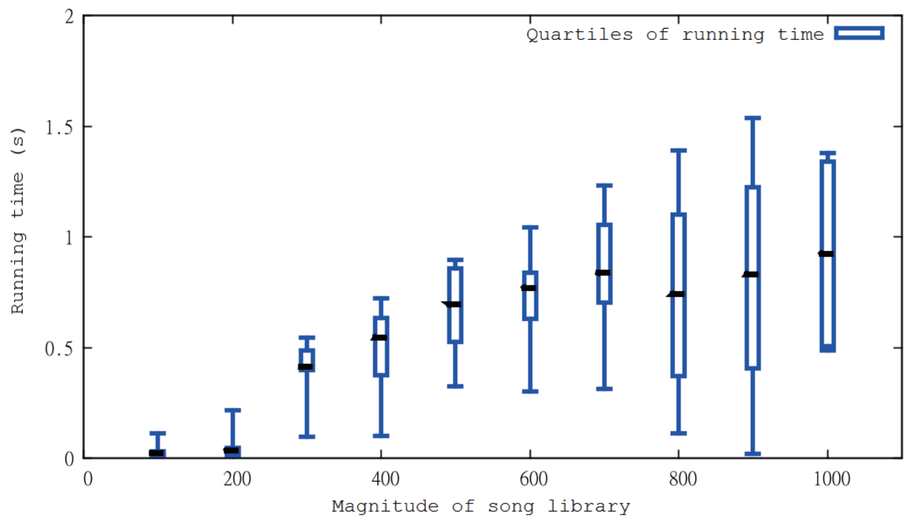


Figure 16: Running time of recommendation function.

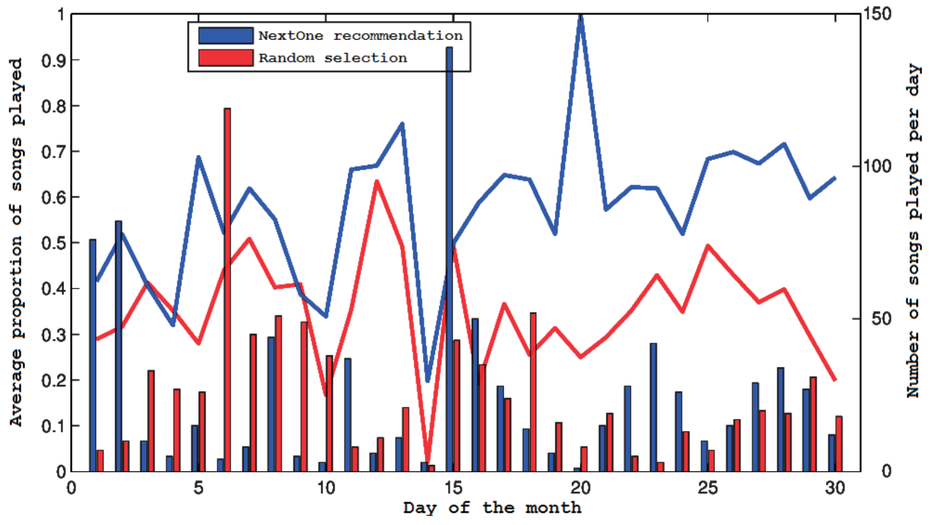


Figure 17: Representing the user logs to express favordness over a month.

“playing proportion”. Moreover, continuous skips are serious negative influence on the user experience, hence the number of continuous skips is used as a measure of user dissatisfaction. A skip is defined as changing to the next track by the user before playing 5% of the length of the current track. Figure 18 shows the distribution of continuous skips using our method and random selection.

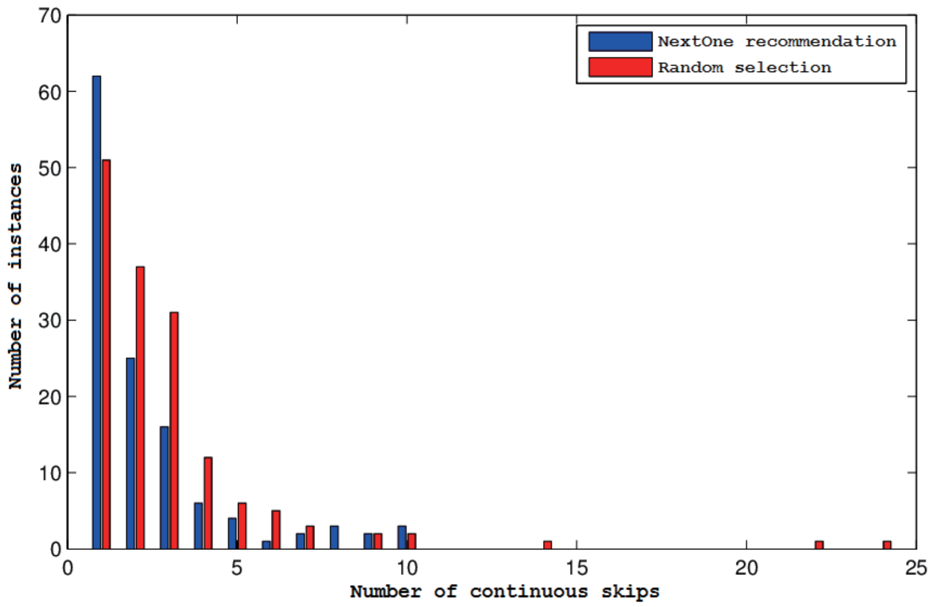


Figure 18: The distribution of continuous skips.

From Figure 17 and 18, we can conclude that the proposed recommendation approach is effective, and surpasses the baseline. Additionally, our approach is able to fit to a user's taste, and adjust the recommendation strategy quickly when then user skips a song.

4.4 Conclusions

This chapter presented a novel approach in recommending songs one by one based on user behavior. The approach considered genre, publish year, freshness, favor and time pattern as factors to recommend songs. The evaluation results demonstrate that the approach is effective. To complement missed genre labels, we proposed classification method to combine sub-classifiers, which are trained by different data sources, and the combined classifier outperforms its sub-classifier, and still beats the classifier trained by all data.

Chapter 5

Adaptive Recommendation

In a listening session, a music recommender would make dozens of recommendations for a user. During the session, the user gives implicit feedbacks to the recommender, i.e., “finish” or “skip”. The implicit feedbacks reflect the user’s taste and current listening context, so they are important and valuable for the recommender to learn the listening pattern of the user and improve the recommendation quality. Furthermore, the recommender is expected to be intelligent to promptly understand users’ intentions behind the feedbacks, and make some changes in the upcoming recommendations.

As the feature evaluation result shows in Chapter 3, Collaborative Filtering (CF) signal is the most important feature among the selected features, so this feature is supposed to be taken into account. When several feedbacks are given in a session, the recommender is expected to update the user-song matrix of the whole recommendation system and retrain the CF model for learning the changed listening pattern. In the real world, the user-song matrix is too large[1] instantly retrain the CF model and make a response in a few seconds.

CF signal is merely one of possible factors, and other factors, like metadata, freshness and context, should be combined together properly as well. The combination model should be adjusted based on the user’s current feedbacks. To be

[1]As announced by Pandora in April 2014, it has 75.3 million active listeners and over 1 million songs.

practical, the expected method must make the learning process as fast as possible, since the adaptive recommendation results should be provided immediately when the feedbacks are given.

In the view of Information Retrieval, we consider a recommender as a ranker, which runs a sophisticated ranking algorithm, and each recommendation is a search, whose the top item of the ranked result is offered as the recommendation result. The search would be like “Play some songs by Rihanna or like her”, “Play some songs for a party”, or “Just play some loved songs”. A user’s profile is automatically created by recent like and dislike songs. The ranker could learn the user’s preference from the profile songs. In a recommendation system, each user is served by a ranker. Even though retraining a ranker spends not too much time, the recommendation system can’t afford to retrain the rankers when every feedback comes. Therefore, unnecessary learning should be avoided and the retraining process must be as fast as possible.

In this paper, we introduce the state-of-art learning to rank approaches by categories in Section 5.1, and the evaluation measures are presented in Section 5.2. Section 5.3 shows how to represent a song for ranking. The fast learning to rank approach is described in Section 5.4. The experimental results are exposed in Section 5.5.

5.1 Learning to rank approaches

As an essential part of a search engine system, learning to rank algorithms have been studied in recent decade, and many algorithms have been proposed.

Generally, these algorithms learn a function $f(q, d)$ to estimate the relevance degree of a document d for a particular query q. The training data contains many pairs of queries and ranked lists. After learning, the function $f(q, d)$ is used to produce a ranked list for a query in a document set.

Basically, these algorithms could be mainly categorized into three classes, i.e., pointwise approach, pairwise approach and listwise approach [Liu, 2009].

5.1.1 Pointwise approach

When people seek a method to use machine learning approaches to solve the problem of ranking, the most straightforward way is converting this problem to a classification or regression problem. This is exactly what the pointwise approach does. The relevance degree of each document is what we are going to predict, though it may be unnecessary since the goal is to generate a ranked list of the documents.

According to different representations of the relevance degree and machine learning approaches used, the pointwise approach can be further divided into three subcategories: classification-based algorithms, regression algorithms and ordinal regression algorithms. For the classification-base algorithms, the output is non-ordered categories, like "Relevance" or "Irrelevance". The output space of regression algorithms contains real-valued relevance scores. The ordinal regression algorithms predict ordered categories.

For the pointwise approach, relevance is absolute and query-independent. However, relevance is query-dependent in practice. An irrelevant document for a popular query might have higher term frequency than a relevant document for a rare query.

5.1.2 Pairwise approach

Essentially, ranking documents focuses on the order between two documents instead of the absolute relevance, like RankBoost [Freund et al., 2003] and RankNet [Burges et al., 2005]. The pairwise approach doesn't care about the relevance degree of each document, but it focuses on the relative order between two documents. Therefore, the pairwise approach is more close to the concept of "ranking" than the pointwise approach.

The pairwise approach usually reduces the ranking problem to a classification problem on document pairs, i.e., to predict which document is preferred. The global goal of the pairwise approach is to minimize the number of incorrect classified pairs. In the perfect case, if the classifier makes correct classification on each document pair, all documents will be correctly ranked.

The pairwise approach ignores the position of pairwise classification errors, so no matter where the error happens, the pairwise approach sees it the same in terms of pairwise classification. Actually, the top positions should be placed more emphasis, so the errors at different positions are supposed to be different in terms of position-discounted evaluation.

5.1.3 Listwise approach

Overcoming the shortcomings of the above two approaches, the listwise approach takes the entire set of documents associated with a query as the input, and learns the rank model by their ground truth labels, like ListNet [Cao et al., 2007], AdaRank [Xu and Li, 2007] and LambdaMART [Wu et al., 2010].

The approach explicitly or implicitly relates the loss function to evaluation measures. The evaluation measures are directly to used as the loss function or applied to generate a loss function, considering continuity, differentiability, convexity, computational efficiency, statistical efficiency, soundness and so on. Because of the relationship between the loss functions and evaluation measures, this approach is also referred to as direct optimization methods. The natural idea is to optimize the measure which is used to evaluate the ranking results.

The fast learning to rank approach is proposed as a listwise approach since this category is close to the reality of ranking.

5.2 Evaluation measures

Despite some learning to rank approaches are able to estimate the relevance degree of a document, the relative position of the document in the ranked list, rather than the absolute relevance degree, reflects the ranking quality. Thus, the comparison between the labeled ranked list and automatically ranked list shows the performance of a learning to rank algorithm. Several popular measures have been developed to evaluate the comparison between two lists from different perspectives.

Precision at n (P@n) is one of the simplest measures, regardless the positions of relevant documents, as shown in the following equation.

$$P@n = \frac{\#\left\{\text{relevant documents in the top } n \text{ results}\right\}}{n} \tag{18}$$

Average Precision at n ($AP@n$) improves P@n by placing high weight on the top relevant documents. The $AP@n$ over all queries in the test set is defined as MAP.

$$AP@n = \frac{\sum_n P@n \cdot I\left\{\text{document } n \text{ is relevant}\right\}}{\#\left\{\text{relevant documents}\right\}} \tag{19}$$

NDCG at n (NDCG@n) evaluates ranking results from the view of information theory, and it places more weight on the top of the list too.

$$NDCG@n = Z_n \sum_{i=1}^{n} \frac{2^{r(i)-1}}{\log\left(1+i\right)}, \tag{20}$$

where $2^{r(i)-1}$ is gain of relevant documents and $\log\left(1+i\right)$ is position discount. Z_n normalizes the value from 0.0 to 1.0.

Moreover, the position of the document with highest relevance degree instead of the list is important for some cases. Best at n (Best@n) focuses on evaluating the importance.

We apply NDCG@n and Best@n to evaluate ranked lists during training a ranking model and evaluate the ranker in the test process.

5.3 Song representation for ranking

A song should be represented by a vector of features, and the vector, as d, is used in the ranking function $f(d, q)$ to estimate the relevance degree of the song. The non-time serial features, like key, mode, tempo and so forth, could be directly used, while time serial features, like beats and tatum, are used as their statistical values.

The memory-based collaborative filtering signal in music recommendation is who listens to what, so, in a straightforward way, it is represented as a user-song matrix, whose cells are the play times of the song by the user. Unfortunately, this way is impractical to represent a song in a real system. If the song is represented as a vector who listens it how many times, the length of the vector would be over one million (This scale of users is common for real music recommendation systems). Furthermore, many songs are listened by a few users, so the vectors of these songs are extremely sparse. Singular Value Decomposition (SVD) is a popular method to deal with sparse high-dimension matrix, and it helps latent semantic indexing perform well in text classification. The computation time for an SVD of an $m \times n$ matrix is $4m^2n+8mn^2+9n^3$. Even though a fast algorithm [Brand, 2006] claims that the time complexity is reduced to O(mnr) for $r \leq \sqrt{\min(m,n)}$, the computation time for a million-scale matrix is still too long.

Essentially, collaborative filtering leverages users' preferences to predict the relevance of an item for a user, so we create a feature space with respect to users' preferences, and locate songs in this feature space. First of all, the users are clustered into k clusters according to preferences. The distance between two users, $\mathrm{Dist}(u_1, u_2)$, is measured by the listening history of them as shown in the following equation.

$$\mathrm{Dist}(u_1, u_2) = \sqrt{\sum\nolimits_{s \in \mathbf{S}} (t_1^s - t_2^s)^2}, \tag{21}$$

where t_1^s is the play time of song s by user u_1 and $\mathbf{S}$ is the entire song collection.

The users in a cluster share similar preferences on music, so the centroid represents these users more or less. A centroid is considered as the origin of a polar coordinate, so a feature space is created by these centroids, and it has k dimensions. In this feature space, a song is represented as a vector $(v_1, v_2, ..., v_k)$, which v_i is set to the distance between centroid c_i and the user who listens to this song.

$$v_i = \frac{\sum\limits_{u \in \mathbf{U}} t_u \cdot \mathrm{Dist}\left(u, c_i\right)}{\sum\limits_{u \in \mathbf{U}} t_u}, \tag{22}$$

where $\mathbf{U}$ is all users who listens to this song.

The vector $(v_1, v_2, ..., v_k)$ of a song, named compressed CF signal, partly represents CF signal, and as shown in Section 5.5, this compressed CF signal performs almost as well as the original CF signal using memory-based CF method.

Finally, the V-dimension vector combined by the compressed CF signal and the content feature are represent songs for ranking.

5.4 Fast learning to rank approach

The ranking results in a music recommendation system are more personalized than that of a query in a search engine, so the ranker should be personalized. A simple but effective way is that one ranker serves one user. Furthermore, in a listening session, the user's feedbacks clue the change of current preference, or assist to drill the preference down. It is expected that the ranker learns the feedbacks and the retrained ranker is applied for the following recommendations. In a real recommendation system, the amount of active users and service requests is so large that the computation time of learning a ranking function is a serious problem.

The listened songs in a period are considered as a list of documents, $\{x_1^u, x_2^u, ..., x_n^u\}$. The relevance of songs, $\{y_1^u, y_2^u, ..., y_n^u\}$, is estimated by listening behavior. These songs and corresponding relevance label generate a document-label pairs, $\{(x_1^u, y_1^u), (x_2^u, y_2^u), ..., (x_n^u, y_n^u)\}$, for a query, i.e., "Currently favorite songs for me". A ranking evaluation measure is applied as the loss function, and the expected ranking function is supposed to rank these songs at the minimum loss according to the loss function.

In order to make learning fast, we apply a series of linear transformations, e.g., scaling and shifting, over the parameters of the ranking function.

$$f^u(x) = \sum_{i=1}^{|V|} a_i^u v_i + b_i^u \tag{23}$$

The ranking function $f^u(x)$ calculates the relevance of song x for user u. a_i^u and b_i^u denote the scaling and shifting operations, respectively.

A close look at the ranking function exposes that v_i is an element of a feature, like CF signal feature, is not independent on other features, so grouping V original features could significantly reduce the search space for the minimize loss and avoid overlearning. The ranking function is therefore improved to the following function.

$$f^u(x) = \sum_{k=1}^{K} \sum_{g(i)=k} a_k^u v_i + b_k^u, \tag{24}$$

where K denotes the number of feature groups. The elements, which belong to a type of feature or different statistic forms, are grouped.

Once the grouping function is given, the important component of this approach is searching scaling and shifting values, $\mathbf{A}$ and $\mathbf{B}$ to minimize the loss $L(f^u)$, namely, one of learning to rank evaluation measures introduced in Section 5.2.

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{B}} \ell(\mathbf{A},\mathbf{B}) = L(f^u) \\ &\text{where} \quad f^u(x) = \sum_{k=1}^{K} \sum_{g(i)=k} a_k^u x_i + b_k^u \end{aligned} \tag{25}$$

It is an optimization problem, so we apply Coordinate Descent to search the minimizer. In each iteration, Coordinate Descent liner searches along one coordinate direction at the current point. Although this optimization algorithm cannot guarantee that the algorithm converges at the global minimizer, the algorithm is so fast to find a local minimizer that it is able to fast response user feedback in a listening session by online training.

5.5 Experiments

5.5.1 Dataset

As introduced in Section 2.3, Taste Profile dataset is a huge user log, including over 48 million `user-song-play count` triplets, over one million unique users and nearly 400 thousand songs. Additionally, the song ID of the Taste Profile

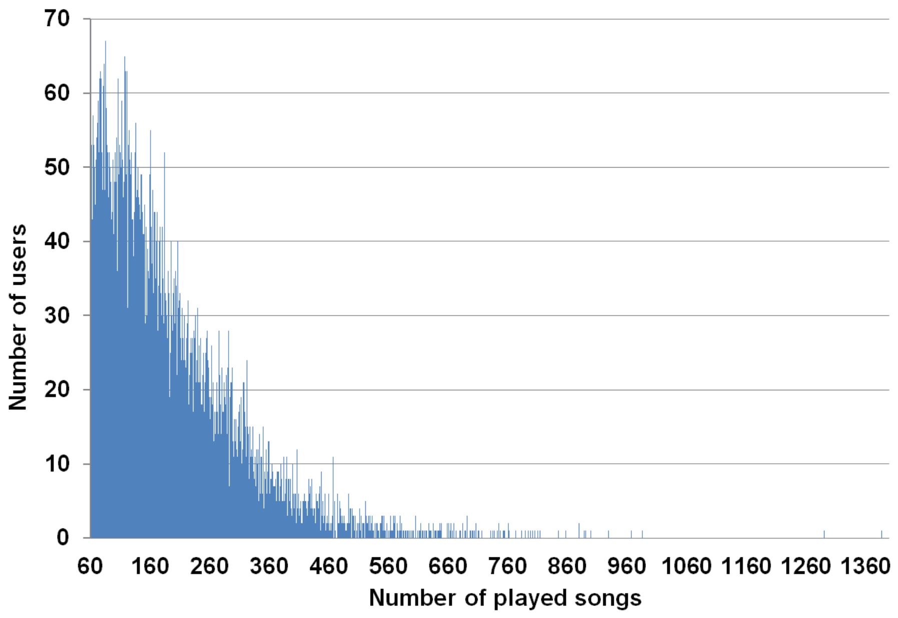


Figure 19: User distribution on unique played songs.

dataset can be mapped to the Million Song Dataset (MSD), so all metadata associated by the ID is available for music representation. Hence, Taste Profile dataset and MSD are used to test and evaluate the proposed approach in this chapter.

In this experiments, we don't attempt to test how the approach performs against "cold start", so we select the users in the Taste Profile dataset who aggregate playtimes are over 500. We further ignore the users who play so much, i.e. 2000 playtimes or more, because we can't verify that these few users are real human rather than robots. Figure 19 demonstrates the distribution of the selected users.

The playtime of a song by a user implies how much the user likes the song, namely, the relevance of the song to the queries searched by the user. Because the Taste Profile dataset doesn't have any time stamps in the user log, so we have to randomly selected 10 songs from n songs listened by a user to generate the test

Feature
Beats confidences (Maximum, average, minimum and variance)
Tatum confidences (Maximum, average, minimum and variance)
Duration
Key
Key confidence
Mode
Mode confidence
Energy
Loudness
Tempo
Artist familiarity
Artist hottness
Song hottness
compressed CF signal (Distances to k centroids in the user taste space)

Table 7: Song representation for ranking.

set of the user, $\mathbf{t}_u$. 10 songs out of the $n-10$ songs are randomly selected with playtimes as the user's feedbacks, $\mathbf{F}_u$, used to retrain the ranking model by the proposed approach. The rest $n-20$ songs as the train set, $\mathbf{T}_u$, are used to train the ranking model.

For each song, we apply the method introduced in Section 5.3 to generate the feature to represent songs including compressed CF signal and the metadata features. Table 7 summarizes the features used in our experiments. k is set to 16.

5.5.2 Experimental results

We apply RankLib[2] to train and test ranking models. The proposed approach is evaluated from three prospectives, i.e., running time, recommendation quality compared to memory-based CF method and ranking performance compared to other learning to rank approaches.

[2]http://people.cs.umass.edu/ vdang/ranklib.html

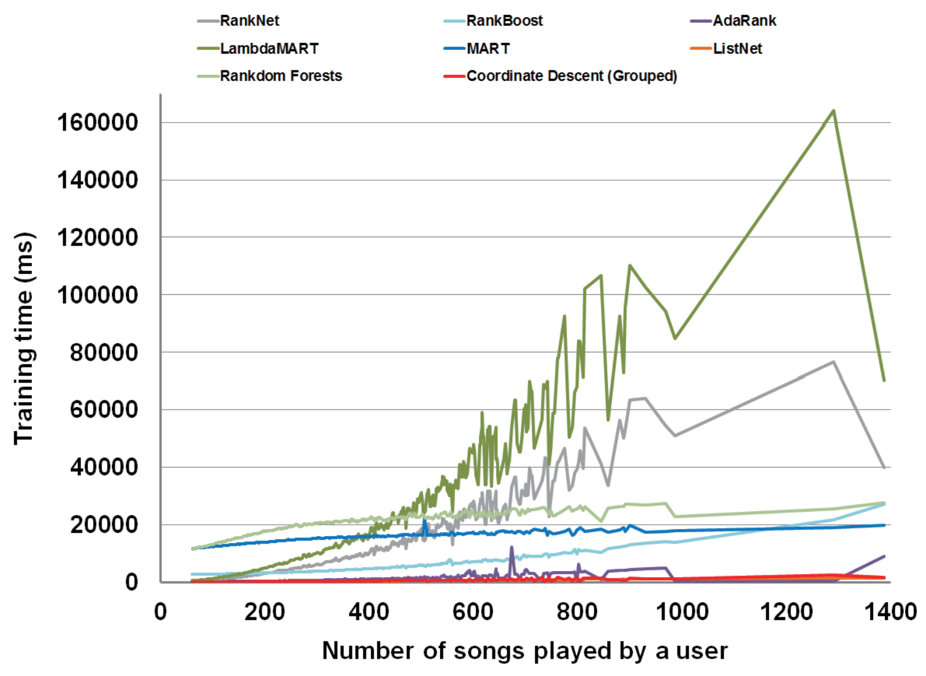


Figure 20: Training time of learning to rank approaches.

Running time

The proposed approach is expected to frequently learn the user feedbacks, so how fast it trains the ranking model is critical in this experiment. We train ranking models for each user by the train set $\mathbf{T}_u$ plus feedbacks $\mathbf{F}_u$ using the our method and other learning to ranking approaches, which cover popular pairwise and listwise approaches, and compare their running time. All loss functions of these training processes are set to NDCG@5. We train these ranking models on a PC, whose the dual-core CPU is 2.4 MHz and the main memory is 2.0 GB.

Figure 20 shows that our method, the red curve, is obviously faster than the most of popular ranking to learn approaches. Even though the number of unique songs played by users goes large, the running time of the proposed approach still keeps short.

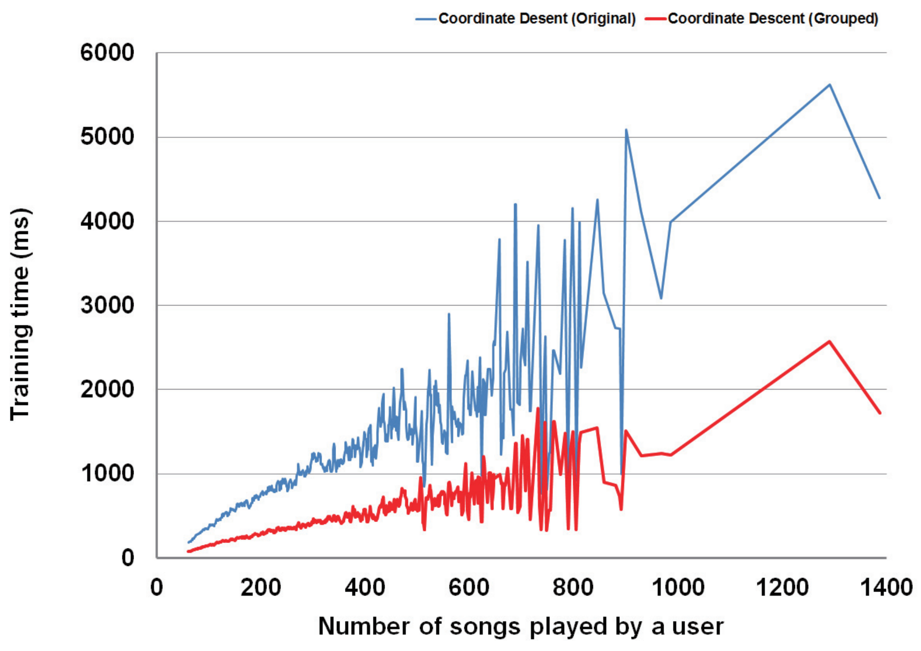


Figure 21: Training time of original and improved coordinate descent methods.

Figure 21 gives a close look at the difference between the training time of original coordinate descent and that of improved coordinate descent. Grouping similar feature elements significantly reduces the searching space for the minimizer, so the coordinate descent would faster converge to the minimizer.

Ranking performance compared to collaborative filtering method

As discussed in Chapter 3, CF signal outperforms other types of metadata in terms of predicting favorite songs, so we compare the proposed learning to rank approach to CF method. We apply memory-based CF method to predict how favorite a song is for a user, namely, the relevance degree, and rank the test songs by the predicted relevance.

In a recommendation music system, it is acceptable for a user to skip a few songs if the recommended songs are not satisfied enough, but the user might not stay and keep listening after many disappointing recommendations. We therefore focus on the top 5 songs of the ranked list, and evaluate the ranking quality by the top songs. The lists ranked by the proposed approach and memory-based collaborative filtering method are evaluated by NDCG@5.

Instead of the original CF signal, the distances to k centroid of clusters in the user taste space is used to represent songs in order to reduce the dimension. Obviously, the compressed CF signal would lose some information of the original signal as the dimension is sharply decreased. The relevance degree of a song is predicted by the relevance of the songs in the training set and the k dimension compressed CF signal as shown in the following equation.

$$f^u\left(x\right) = \sum_{s\in\mathbf{T}_u} \frac{y_s^u}{dist\left(s,x\right)}, \tag{26}$$

where y_s^u denotes the relevance degree of a training song s for the user u, and $dist\left(s,x\right)$ is the Euclidian distance between the song S and the song x according to the compressed CF signal. 10 period moving average is used to relief the big fluctuation of the result curves due to the small number of user cases.

As shown in Figure 22 and 23, the ranking results used by the compressed CF signal are almost as good as that used by the original CF signal. It is proved that the compressed CF signal keeps main information of the original one. Moreover, the proposed approach obviously surpasses the memory-based CF method.

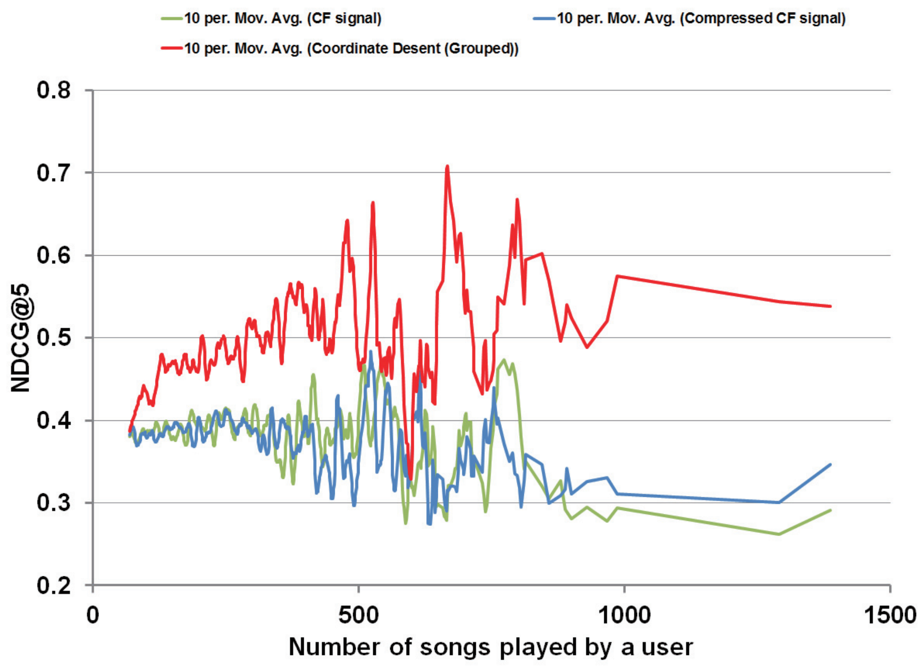


Figure 22: NDCG@5 results of the proposed approach and CF methods.

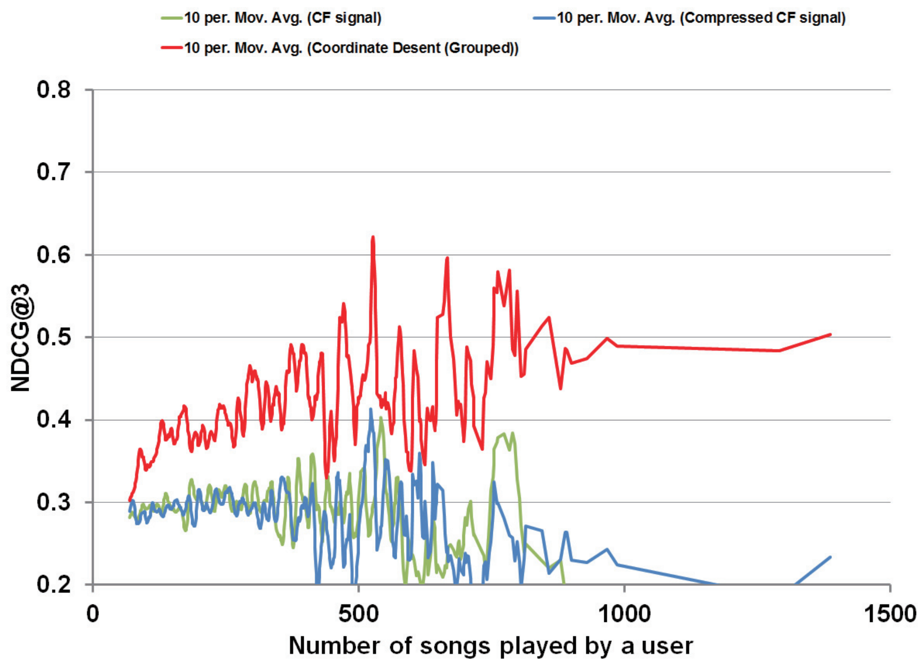


Figure 23: NDCG@3 results of the proposed approach and CF methods.

Ranking performance compared to other learning to rank approaches

Since the rankers are not trained frequently in a ranking system due to high time complexity, the training set $\mathbf{T}_u$, excluding the feedback set $\mathbf{F}_u$, is used to train the ranking models. Because the proposed approach is able to fast learn a user's feedbacks, the ranking model is trained by $\mathbf{T}_u$ plus $\mathbf{F}_u$. The trained models predict the relevance degree of the songs in the test set $\mathbf{t}_u$, and rank the songs by the relevance. The predicted ranked lists are evaluated by a evaluation measure.

10 period moving average is used to smooth the curve, and the NDCG@5 evaluation results are shown in Figure 24. The evaluation results show that the proposed approach is encouraging, and it performs better than most of other popular learning to rank approaches. Furthermore, at the small number of songs played by a user, the advance of the proposed approach is more significant, because the feedback plays more important supplement when a user listens to not too many songs.

Furthermore, we focus on the smaller range of the top of the ranked list, because we expect that a outstanding learning to rank approach is able to rank favorite songs at the top of the ranked list. Even though the highest favorite song is not ranked at the top, placing it at the second or the third position is not bad. Thus, we apply NDCG@3 to evaluate the learning to rank approaches again, and the results are shown in Figure 25.

Additionally, these learning to rank approaches are evaluated according to the predicted position of the best one in the test set. We apply Best@5 to evaluate

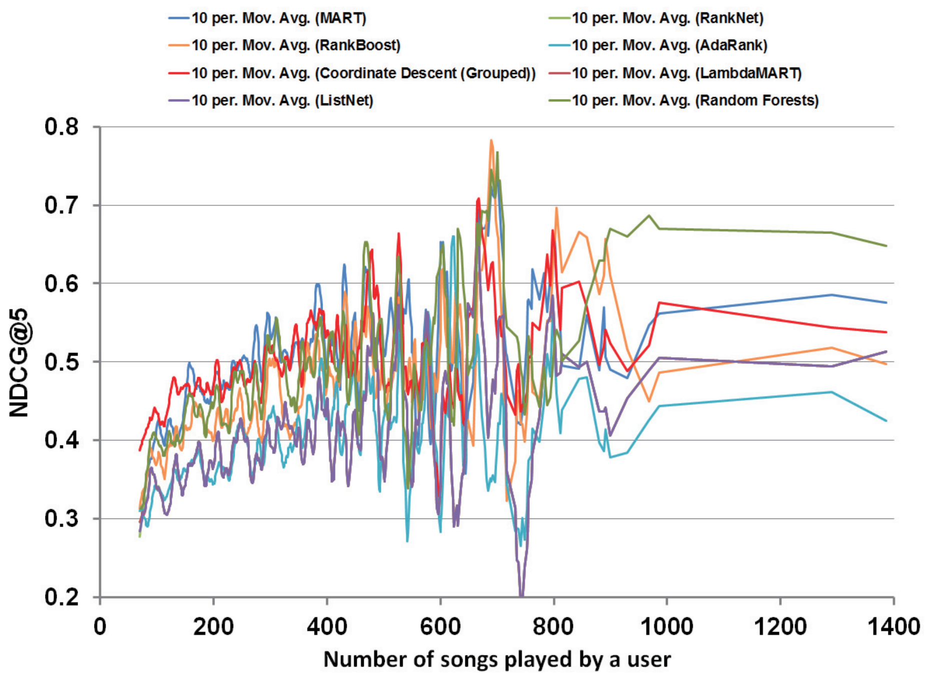


Figure 24: NDCG@5 results of learning to rank approaches.

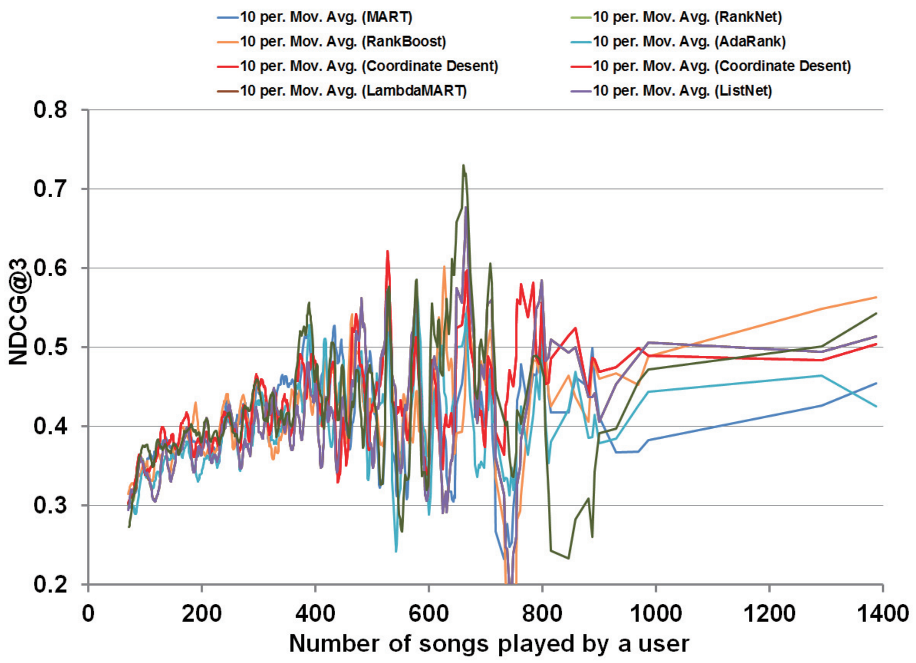


Figure 25: NDCG@3 results of learning to rank approaches.

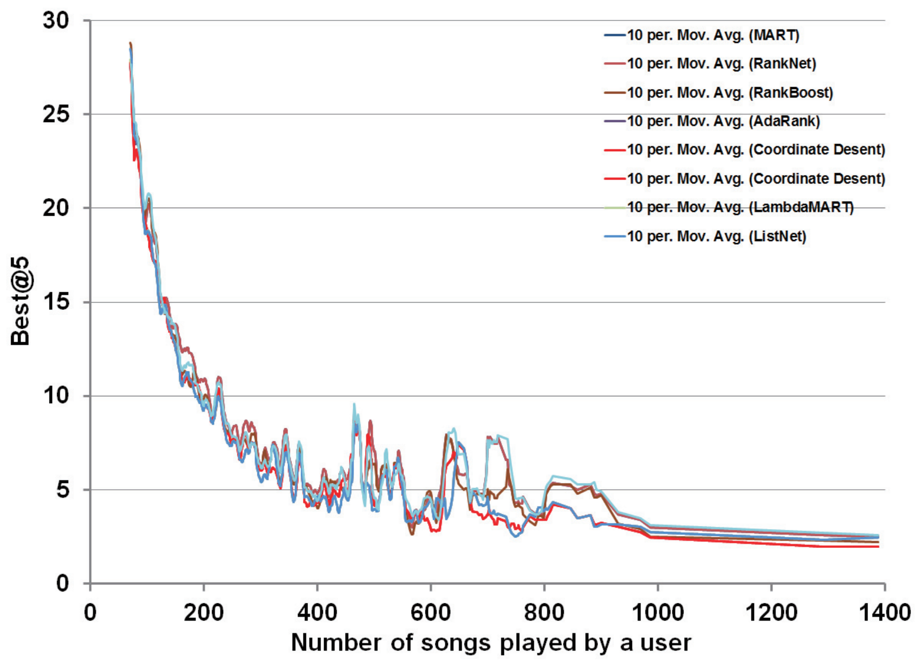


Figure 26: Best@5 results of learning to rank approaches.

these approaches. Low score of the Best@5 means that the position of the best song is close to the top of the ranked list. Figure 26 shows that the proposed approach is promising too.

5.6 Conclusions and discussion

To our knowledge, we first consider music recommendation problem as a ranking problem, and the experimental results support that this consideration works very well. The compressed CF signal is so informatively that it produces a comparative performance to the original CF signal. Additionally, grouping related feature elements significantly reduces the dimension of the space to search the minimizer, and contributes the coordinate descent approach to converge fast enough to frequently learn a user's feedbacks. The experimental results show that the proposed approach is able to promptly learn a user's feedbacks and give satisfactory recommendations. As a result, the learning speed and the ranking performance make the proposed method practical in the real world.

Due to the limitation of the data set, the ranking model has to ignore the information about when a user listens to the song. If the data set contained a timestamp for each playing behavior, freshness, listening pattern and other features about context could be added into the ranking model. Furthermore, recent feedbacks would be more important than old ones, so the recent feedbacks are supposed to be placed more emphasis, and some old ones should be removed from the retraining set.

Chapter 6

Music Recommendation for Implicit User Groups

Recently, there are several online music applications providing personal music service, like Pandora, Last.fm, Spotify and so forth. Instead of these recommenders, which recommend music to individuals, wahwah provides a new service: "co-listening". This service allows a user to create a personal radio station with an activity label, and broadcast the radio station in public. Other users can filter public radio stations by activities or other attributes, and enter a radio station to listen to music together. It makes enjoying music with friends or strangers doing the same activity possible. Defined simply as a set of co-listening users, a *user group* can be formed on a friendship basis, e.g., friends who listen to music in a party, or on an ephemeral basis, e.g., random users in gyms for a workout and listening together. Regardless how the group is formed, there are two major challenges for recommending to a group in comparison to individual user recommendations [Jameson and Smyth, 2007].

The first challenge is how to define group recommendation. The usual goal in individual music recommendation is to retrieve songs with the highest scores, also referred to as relevance or user's expected rating, estimated by a given recommendation strategy. In group music recommendation, however, a song may have different relevance scores to different group members, so this disagreement among members must be resolved.

A group could be temporarily formed, and there is no rating history and feedback data of the group. The profiles of the group members may however be available, so how to aggregate the members' ratings to the single group rating is a main problem. Existing methods can be classified into two types [Amer-Yahia et al., 2009]: *preference aggregation*, which aggregates group members' prior ratings into a single virtual user profile and makes recommendations to that user; *score aggregation*, which computes each member's individual recommendations and merges them to produce a single list for the group, where the score of each item is aggregated from individual recommendations. The score aggregation approach has better flexibility [O'Connor et al., 2002] [Amer-Yahia et al., 2009] and more potentiality for efficiency improvement. Therefore, we take the score aggregation in this paper.

There are a number of goals that may be desirable in any given situation, e.g., total satisfaction, fairness, comprehensibility and so forth. Two main score aggregation functions have been proposed so far [Jameson and Smyth, 2007]: *maximizing average satisfaction* and *minimizing misery*. The former aims to maximize the average of group members' scores for an item while the latter aims at maximizing the lowest score among all group members. Intuitively, the average aggregation for expected ratings may result in a not high score for songs which are loved by the majority of group members but highly disliked by others. Taking the minimizing misery aggregation would underestimate a song's score which is loved by every member except one. As the music preferences of users are so much personal and the cost of changing a group is so small, the aggregated rating is expected to represent the major of members while the minor others who disagree with the rating maybe should join other groups. Therefore, we evaluate the satisfaction of a song for a group by the major opinion, named *maximizing majority satisfaction*.

How to efficiently make group recommendation give a consensus function is the second challenge. If the recommendation scores from individual relevance lists (a list of items sorted by their scores for each user) are available, a weighted-based summation algorithm is able to well support the consensus function. Considering the relevance and disagreement of group members, initializing the weight would be a time-consuming task. When a user enters or leaves from a group, the relevance and disagreement may be changed, and how to update the weight is also a efficiency-related problem.

When a recommender system makes a recommendation for a group, the group members could vote by feet, namely, staying in the group or leaving from it anytime. For example, if a user enters the group whose members have similar tastes to the user, the user would possible feel comfortable to stay in the group and listen to music with others together. Otherwise, the user would like to leave from the group, and explore other groups. As a result, the group recommendation is meaningless for the leaving user. Hence, the recommender should cater the majority's tastes because they are more likely to stay in the group and listen to the recommended song.

Therefore, our endeavor is to estimate the probability that a group member stay in the group, and use the probability to aggregate the individual ratings. The probability is estimated by the music relationship between users, which is calculated by what songs they played and how similar these songs are.

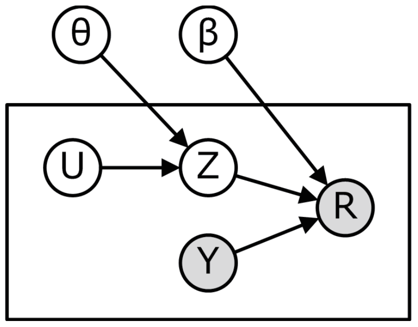
As we discussed in Chapter 3, collaborative filtering signal has better performance than other content-based features, so, in this chapter, we apply collaborative filtering technique to predict ratings of members in a group.

6.1 Group music recommendation method

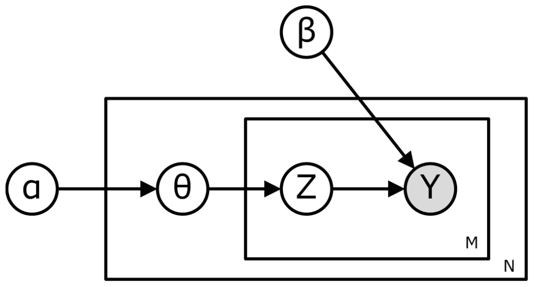
6.1.1 Rating prediction model

The rating-based collaborative filtering techniques received significant attention in the past. The goal is to predict the user's rating for the item, given a particular user and item. Ideal personalized recommender can predict the ratings for any user by existing ratings to all items that the user has not rated, and recommend items with the highest predicted scores. Rating prediction can be incorporated with content information [Yoshii et al., 2006].

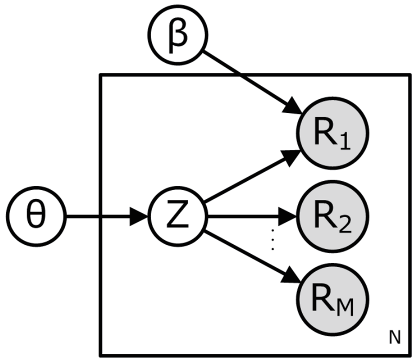
There are a handful of probabilistic models to calculate the distribution over ratings, and predict ratings, as shown in Figure 27 [Marlin, 2003]. In all models, U is a user, Y is an item, Z is a user attitude, R is a rating score, and α, β and θ are parameters. These models are to maximize the advantage from the expressive power of existing ratings. Both the aspect model [Hofmann, 2001] and the multinomial mixture model have an intuitive appeal as latent variable models. They decompose user preferences profiles into a set of typical preference patterns, and the latent



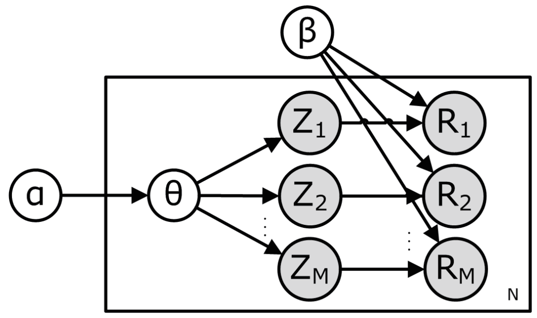
(a) Aspect Model



(b) LDA Model



(c) Multinomial Mixture Model



(d) URP Model

Figure 27: Probabilistic models for rating prediction.

variables are casually referred to as *user attitudes.* Latent Dirichlet Allocation (L-DA) [Blei et al., 2003] is one of generative latent variable models for discrete data. It is designed with co-occurrence data, like user-item pairs, and used to generate recommendations. However, it cannot be used to infer a distribution over ratings, or to predict the scores. User Rating Profiles (URP) model [Marlin, 2003] combines the intuitive appeal of the multinomial mixture model and the aspect model using generative semantics of LDA. URP model outperforms the aspect model and the multinomial mixture model for rating prediction tasks [Marlin, 2003].

We use URP model to predict the probability, $p\left(R_y = v|\mathbf{r}^u\right)$, which is the probability of rating score v for item y given a member rating profile $\mathbf{r}^u$.

6.1.2 Estimation for the probability of memberships

A user may stay in a group and listen to music with other group members who have similar tastes. The probability of staying in a group therefore depends on how much the group members are similar to the user in terms of music preference.

If two users like or dislike similar songs, they might have similar preference. On the other hand, if they like or dislike entirely different songs, or a user likes the song which the other one dislikes, the similarity between these two users should be low. The similarity of users is thereby related to the song similarity and their opinions on these songs. Table 8 summarizes the four cases and the expected distance between two users, if there are two ratings for two songs assigned by two users.

Difference between two ratings	Distance between two songs	Expected distance between two users
Low	Low	Low
Low	High	High
High	Low	High
High	High	Low

Table 8: Four cases and corresponding distances of users.

If user u_i rated a set of songs $\mathbf{Y}_i$, and user u_j rated a set of songs $\mathbf{Y}_j$, the distance between user u_i and user u_j, $dist\,(u_i, u_j)$, is measured by Equation 27, referring to hyperbolic function.

$$\text{if } |\mathbf{Y}_i| \leq |\mathbf{Y}_j|$$

$$dist\,(u_i, u_j) = \frac{\sum_{y_m \in \mathbf{Y}_i} \left| \lambda \min_{y_n \in \mathbf{Y}_j} dist(y_m, y_n)^2 - \eta \left[\left(r^{u_i}_{y_m} - r^{u_j}_{y_n} \right) / V \right]^2 \right|}{|\mathbf{Y}_i|}, \tag{27}$$

$$\text{otherwise}$$

$$dist\,(u_i, u_j) = dist\,(u_j, u_i)\,,$$

where $dist\,(y_m, y_n)$ is the distance between song y_m and song y_n, from 0.0 to 1.0. $\left(r^{u_i}_{y_m} - r^{u_j}_{y_n} \right) / V$ is the difference between two ratings, from 0.0 to 1.0. λ and η are two regulatory factors.

We assume that the distance between two songs, $dist\,(y_m, y_n)$, is reflexive, symmetric and transitive, and this assumption will be proved later. Because $dist\,(y_m, y_m)$ is reflexive, i.e. $dist\,(y_m, y_m) = 0$, $dist\,(u_i, u_i)$ is consequentially equal to 0, and the measurement is reflexive. Obviously, $dist\,(u_i, u_j) = dist\,(u_j, u_i)$ proves that the measurement is symmetric. Because ratings assigned by users are arbitrary, we cannot prove the measurement be transitive in mathematics. We randomly select 1,000 users, and calculate the distance between them. Over 80% of cases are transitive. Furthermore, the measurement well supports the four cases as shown in Table 8.

The song distance $dist\left(y_m, y_n\right)$ is measured by song metadata and acoustic features. For sequential features, like pitch, Dynamic Time Warping (DTW) algorithm, which is able to find the optimal alignment between two time series, is applied to estimate the difference between two songs in terms of sequential features. A quadratic running time and high space complexity of DTW algorithm however limits the scale of time series data. Hence, we apply FastDTW [Salvador and Chan, 2007] to accelerate the similarity computation. The distance in respect of non-sequential features or metadata is measured by Euclidean distance. $dist\left(y_m, y_n\right)$ is the combination of these distances. Since Euclidean distance and Dynamic Time Warping are reflexive, symmetric, and transitive, the weighted combination is consequentially reflexive, symmetric, and transitive too.

The probability of user u staying in group G is estimated by Equation 28.

$$p\left(u|G\right) \sim \frac{1}{\sum\limits_{u' \in G, u' \neq u} dist\left(u, u'\right)} \tag{28}$$

The result, combined $p\left(u|G\right)$ and $p\left(R_y = v|\mathbf{r}^u\right)$, is applied to estimate the distribution over ratings for item y assigned by group G. The rating score v which $p\left(R_y = v|G\right)$ is maximum is treated as the single value result for other possible applications, like group music recommendation.

All user distances and the trained URP model is supposed to be stored in memory, and reused in other predictions. When a group is reorganized, the additional computation is not too much because of the reuse.

6.2 Experiments

6.2.1 Experiment settings

Compared to other available dataset, aforementioned in Section 2.3, we use Taste Profile Dataset to evaluate the proposed approach. In the Taste Profile Dataset, a great number of songs are listened by over 10 users, and they are treated as the songs listened by user groups in the following experiments. For each song, we randomly select 10 users among who listened to the song, and convert the paly count of each user to the rating score assigned by the user, as the ground truth in the experiments. In our experiments, these selected users are assumed that they don't listen to the song. The reason why we select 10 users instead of all users is that collaborative filtering method predicts the individual rating score for a song by the ratings for the song assigned by unselected users.

maximizing average satisfaction and *maximizing majority satisfaction* metrics are used to evaluate the satisfaction of a group user. The former uses average scores of individual ratings to estimate the group rating score while the latter treats the rating score of the majority as the group score. We apply Affinity Propagation clustering method [Frey and Dueck, 2007] to cluster the individual scores. This method doesn't need to predefine the number of clusters, and all that it needs is a similarity preference, like threshold. In practice, the default threshold is set to

	Our method	Memory-base CF
RMSE	**0.8791**	0.9171
Correlation	**0.3829**	0.2129

Table 9: Comparison result for maximizing majority satisfaction.

	Our method	Memory-base CF
RMSE	0.6921	**0.6380**
Correlation	**0.2753**	0.0909

Table 10: Comparison result for maximizing average satisfaction.

the median of all distance. We use this method to cluster individual rating scores in a group, and get a set of clusters $\{c_1, c_2, ..., c_n\}$ sorting by the cluster size. The group score is given by the following equation.

$$\begin{aligned} r_y^g &= \frac{\sum_{u \in c_i, i=1}^{l} r_y^u}{\sum_{i=1}^{l} |c_i|}, \\ \sum_{i=1}^{l} |c_i| &\geq \lambda \sum_{i=1}^{n} |c_i| \geq \sum_{i=1}^{l-1} |c_i|, \end{aligned} \tag{29}$$

where λ is the majority proportion, and it is set to 0.7.

In order to evaluate the proposed method, we implemented memory-based collaborative filtering method to predict individual rating scores, and use the mean score of the individual scores to predict the group rating score.

6.2.2 Experimental results

We use Root Mean Square Error (RMSE) to measure the difference between predicted scores and true scores. Obviously, the method with smaller RMSE value surpasses the other one. If the scores predicted by method A are always equal to the true scores +2.0, and method B randomly predicts scores in the true scores ± 1.0, method A would be better than B because the scores predicted by method A

strictly depend on the ground truth, and it is easy to convert the predicted scores to the true scores by adding a bias. As a result, we use Correlation between the predicted scores and the ground truth to evaluate the strength of the dependence between these two series of scores. The comparisons between our method and the memory-base collaborative filtering technique are shown in Table 9 and 10.

These two tables show the comparison results based on two different aggregation methods, i.e., maximizing majority satisfaction and maximizing average satisfaction. These two aggregation methods are used to generate the ground truth by aggregating the true individual scores. As shown in Table 9 and 10, our method generally outperforms to the memory-based collaborative filtering method.

6.3 Conclusion

This chapter proposes a method to predict rating scores for songs assigned by a group of users. Individual rating scores are predicted by URP model first, and these scores are aggregated by staying probability of the group members, which is estimated by the preference similarity among the group members. We implement the memory-based collaborative filtering method, and compare to the proposed method under a large user log. As shown in the experimental results, the proposed method has a better performance to predict the group rating scores than the memory-based collaborative filtering method.

Chapter 7

Conclusions

This dissertation introduces and discusses several problems on music recommendation. The feature importance for music recommendation is analyzed by experiments in statistics. As shown in the experimental results, collaborative filtering signal is the most important features for favorite song detection to make recommendations.

Bringing favorite songs is the important part of a music recommender, but it is not enough to make users satisfied. A comparison experiment based on a music recommendation system shows that the proposed recommendation algorithm improves the recommendation quality on favorite songs. Freshness, time pattern, publish year, genre and favor support the recommender to make qualified recommendations.

Based on classifier authority and classification confidence, the proposed combination method gently makes several trained classifier work together in different views to classify music genre. The combined classifier performs high classification accuracy, and it is further better than the classifier trained by concatenated vector from different data sources.

The compressed collaborative filtering signal along with metadata of songs effectively represent music, and the grouped features are used to train rankers by coordinate descent. This improved approach is faster than other learning to rank approaches to learn the user's preferences. It provides adaptive music recommendation, and the recommendation quality is promising. In the contrast experiment, the proposed approach outperforms other approaches and memory-based collaborative filtering technique.

We consider the members of a user group as the results after probability events, namely, staying or leaving. The recommender for a user group estimates the probabilities that the members stay in the group for the next recommendation, and it places more consideration on the preferences of who will stay in the group. The recommendation quality is encouraging in our experiments.

REFERENCES

[Amer-Yahia et al., 2009] Amer-Yahia, S., Roy, S. B., Chawlat, A., Das, G., and Yu, C. (2009). Group recommendation: semantics and efficiency. *Proc. VLDB Endow.*

[Anglade et al., 2007] Anglade, A., Tiemann, M., and Vignoli, F. (2007). Virtual communities for creating shared music channels. In *Proceedings of 8th International Conference on Music Information Retrieval*, Vienna, Austria.

[Baccigalupo et al., 2008] Baccigalupo, C., Plaza, E., and Donaldson, J. (2008). Uncovering affinity of artists to multiple genres from social behaviour data. In *Proceedings of International Conference on Music Information Retrieval*, Philadelphia, Pennsylvania, USA.

[Barrington et al., 2009] Barrington, L., Oda, R., and Lanckriet, G. R. (2009). Smarter than genius? human evaluation of music recommender systems. In *Proceedings of International Conference on Music Information Retrieval*, Kobe, Japan.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*

[Bogdanov and Herrera, 2011] Bogdanov, D. and Herrera, P. (2011). How much metadata do we need in music recommendation? a subjective evaluation using preference sets. In *Proceedings of 12th International Conference on Music Information Retrieval*, Miami, Florida, USA.

[Bosteels et al., 2009] Bosteels, K., Pampalk, E., and Kerre, E. E. (2009). Evaluating and analysing dynamic playlist generation heuristics using radio logs and fuzzy set theory. In *Proceedings of International Conference on Music Information Retrieval*, Kobe, Japan.

[Box and Pierce, 1970] Box, G. E. P. and Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Jour. of the American Statistical Association.*

[Brand, 2006] Brand, M. (2006). Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications.*

[Burges et al., 2005] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of International Conference on Machine Learning.*

[Cai et al., 2007] Cai, R., Zhang, C., Wang, C., Zhang, L., and Ma, W. (2007). Musicsense: Contextual music recommendation using emotional allocation modeling. In *ACM International Conference On Multimedia*, pages 553–556.

[Cano et al., 2005] Cano, P., Koppenberger, M., and Wack, N. (2005). An industrial-strength content-based music recommendation system. In *28th Annual International ACM SIGIR Conference*.

[Cao et al., 2007] Cao, Z., Qin, T., Liu, T., Tsai, M., and Li, H. (2007). Learning to rank: From pairwise approach to listwise approach. In *Proceedings of International Conference on Machine Learning*.

[Cataltepe and Altinel, 2007] Cataltepe, Z. and Altinel, B. (2007). Music recommendation based on adaptive feature and user grouping. In *22nd international symposium on Computer and information sciences.*, Ankara, Turkey.

[Celma, 2010] Celma, O. (2010). *Music recommendation and discovery*. Springer.

[Celma et al., 2005] Celma, O., Ramrez, M., and Herrera, P. (2005). Foafing the music: A music recommendation system based on rss feeds and user preferences. In *Proceedings of 6th International Conference on Music Information Retrieval*, London, UK.

[Chai and Vercoe, 2000] Chai, P. and Vercoe, B. (2000). Using user models in music information retrieval systems. In *Proceedings of 1st International Conference on Music Information Retrieval*, Plymouth, MA, USA.

[Chao et al., 2005] Chao, D. L., Balthrop, J., and Forrest, S. (2005). Adaptive radio: achieving consensus using negative preferences. In *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, Sanibel Island, Florida, USA.

[Chordia et al., 2008] Chordia, P., Godfrey, M., and Rae, A. (2008). Extending content-based recommendation: The case of indian classical music. In *Proceedings of International Conference on Music Information Retrieval*, Philadelphia, Pennsylvania, USA.

[Cohen and Fan, 2000] Cohen, W. W. and Fan, W. (2000). Web-collaborative filtering: Recommending music by crawling the web. *Computer Network*, 33:685–698.

[Crossen et al., 2002] Crossen, A., Budzik, J., and Hammond, K. J. (2002). Flytrap: intelligent group music recommendation. In *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, San Francisco, California, USA.

[Cunningham et al., 2006] Cunningham, S. J., Bainbridge, D., and Falconer, A. (2006). “more of an art than a science”: Supporting the creation of playlists and mixes. In *Proceedings of 7th International Conference on Music Information Retrieval*, Victoria, Canada.

[Cunningham and Nichols, 2009] Cunningham, S. J. and Nichols, D. M. (2009). Exploring social music behaviour: An investigation of music selection at parties. In *Proceedings of International Conference on Music Information Retrieval*, Kobe, Japan.

[Donaldson, 2007a] Donaldson, J. (2007a). A hybrid social-acoustic recommendation system for popular music. In *ACM Recommender Systems*, Minnesota.

[Donaldson, 2007b] Donaldson, J. (2007b). Music recommendation mapping and interface based on structural network entropy. In *Proceedings of 8th International Conference on Music Information Retrieval*, Vienna, Austria.

[Ebbinghaus, 1913] Ebbinghaus, H. (1913). *Memory: A Contribution to Experimental Psychology*. Columbia University, New York.

[Fields et al., 2008] Fields, B., Rhodes, C., Casey, M., and Jacobson, K. (2008). Social playlists and bottleneck measurements: Exploiting musician social graphs using content-based dissimilarity and pairwise maximum flow values. In *Proceedings of International Conference on Music Information Retrieval*, Philadelphia, Pennsylvania, USA.

[Flexer et al., 2008] Flexer, A., Schnitzer, D., Gasser, M., and Widmer, G. (2008). Playlist generation using start and end songs. In *Proceedings of International Conference on Music Information Retrieval*, Philadelphia, Pennsylvania, USA.

[Forman, 2003] Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*

[Freund et al., 2003] Freund, Y., Iyer, R., Schapire, R., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research*.

[Frey and Dueck, 2007] Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *Science*.

[Gulik and Vignoli, 2005] Gulik, R. v. and Vignoli, F. (2005). Visual playlist generation on the artist map. In *Proceedings of 6th International Conference on Music Information Retrieval*, London, UK.

[Hoashi et al., 2003] Hoashi, K., Matsumoto, K., and Inoue, N. (2003). Personalization of user profiles for content-based music retrieval based on relevance feedback. In *Proceedings of the eleventh ACM international conference on Multimedia*, Berkeley, CA, USA.

[Hofmann, 2001] Hofmann, T. (2001). Learning what people (don't) want. In *Machine Learning: ECML 2001*.

[Hu et al., 2006] Hu, X., Downie, J. S., and Ehmann, A. F. (2006). Exploiting recommended usage metadata: Exploratory analyses. In *Proceedings of 7th International Conference on Music Information Retrieval*, Victoria, Canada.

[Hu and Ogihara, 2011] Hu, Y. and Ogihara, M. (2011). Nextone player: A music recommendation system based on user behavior. In *Proceedings of 12th International Conference on Music Information Retrieval*, Miami, Florida, USA.

[Hu and Ogihara, 2012] Hu, Y. and Ogihara, M. (2012). Genre classification for million song dataset using confidence-based classifiers combination. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*.

[Hu et al., 2013] Hu, Y., Wang, D., and Ogihara, M. (2013). Evaluation on feature importance for favorite song detection. In *Proceedings of the 14th International Music Information Retrieval Conference*.

[Jameson and Smyth, 2007] Jameson, A. and Smyth, B. (2007). Recommendation to groups. In *The Adaptive Web*. Springer Berlin Heidelberg.

[Kamalzadeh et al., 2012] Kamalzadeh, M., Baur, D., and Moller, T. (2012). A survey on music listening and management behaviours. In *Proceedings of 13th International Conference on Music Information Retrieval*, Porto, Portugal.

[Konstas et al., 2009] Konstas, I., Stathopoulos, V., and Jose, J. M. (2009). On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, Boston, MA, USA.

[Laplante, 2011] Laplante, A. (2011). Social capital and music discovery: An examination of the ties through which late adolescents discover new music. In *Proceedings of 12th International Conference on Music Information Retrieval*, Miami, Florida, USA.

[Lee, 2010] Lee, C.-H. (2010). Learning to combine discriminative classifiers: confidence based. In *Proceedings of the 16th ACM SIGKDD*, KDD '10, pages 743–752, New York, USA.

[Lee et al., 2011] Lee, J. H., Bare, B., and Meek, G. (2011). How similar is too similar?: Exploring users' perceptions of similarity in playlist evaluation. In *Proceedings of 12th International Conference on Music Information Retrieval*, Miami, Florida, USA.

[Lee and Waterman, 2012] Lee, J. H. and Waterman, N. M. (2012). Understanding user requirements for music information services. In *Proceedings of 13th International Conference on Music Information Retrieval*, Porto, Portugal.

[Liu et al., 2009] Liu, N., Lai, S., Chen, C., and Hsieh, S. (2009). Adaptive music recommendation based on user behavior in time slot. *International Journal of Computer Science and Network Security*, 9:219–227.

[Liu, 2009] Liu, T.-Y. (2009). *Learning to Rank for Information Retrieval*. Springer.

[Logan, 2002] Logan, B. (2002). Content-based playlist generation: Exploratory experiments. In *Proceedings of 3rd International Conference on Music Information Retrieval*, Paris, France.

[Logan, 2004] Logan, B. (2004). Music recommendation from song sets. In *Proceedings of 5th International Conference on Music Information Retrieval*, Barcelona, Spain.

[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 *Proceedings of International Conference on Music Information Retrieval*, Philadelphia, Pennsylvania, USA.

[Maillet et al., 2009] Maillet, F., Eck, D., Desjardins, G., and Lamere, P. (2009). Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of International Conference on Music Information Retrieval*, Kobe, Japan.

[Mak et al., 2010] Mak, C. M., Lee, T., Senapati, S., Yeung, Y. T., and Lam, W. K. (2010). Similarity measures for chinese pop music based on low-level audio signal attributes. In *Proceedings of 11th International Conference on Music Information Retrieval*, Utrecht, Netherlands.

[Marlin, 2003] Marlin, B. (2003). Modeling user rating profiles for collaborative filtering. *Advances in Neural Information Processing Systems*, 16.

[Masahiro et al., 2008] Masahiro, N., Takaesu, H., Demachi, H., Oono, M., and Saito, H. (2008). Development of an automatic music selection system based on runners step frequency. In *Proceedings of 9th International Conference on Music Information Retrieval*, Philadelphia, Pennsylvania, USA.

[McCarthy and Anagnost, 1998] McCarthy, J. F. and Anagnost, T. D. (1998). Musicfx: an arbiter of group preferences for computer supported collaborative workouts. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, Seattle, Washington, USA.

[McEnnis and Cunningham, 2007] McEnnis, D. and Cunningham, S. J. (2007). Sociology and music recommendation systems. In *Proceedings of 8th International Conference on Music Information Retrieval*, Vienna, Austria.

[McFee et al., 2010] McFee, B., Barrington, L., and Lanckriet, G. R. (2010). Learning similarity from collaborative filters. In *Proceedings of 11th International Conference on Music Information Retrieval*, Utrecht, Netherlands.

[McFee and Lanckriet, 2009] McFee, B. and Lanckriet, G. R. (2009). Heterogeneous embedding for subjective artist similarity. In *Proceedings of International Conference on Music Information Retrieval*, Kobe, Japan.

[McFee and Lanckriet, 2011] McFee, B. and Lanckriet, G. R. (2011). The natural language of playlists. In *Proceedings of 12th International Conference on Music Information Retrieval*, Miami, Florida, USA.

[McKay et al., 2010] McKay, C., Burgoyne, J. A., Hockman, J., Smith, J. B. L., Vigliensoni, G., and Fujinaga, I. (2010). Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features. In *Proceedings of the 11th ISMIR*, ISMIR '10, pages 213–718, Utrecht, Netherlands.

[Miller et al., 2010] Miller, S., Reimer, P., Ness, S. R., and Tzanetakis, G. (2010). Geoshuffle: Location-aware, content-based music browsing using self-organizing tag clouds. In *Proceedings of 11th International Conference on Music Information Retrieval*, Utrecht, Netherlands.

[Moore et al., 2012] Moore, J. L., Chen, S., Joachims, T., and Turnbull, D. (2012). Learning to embed songs and tags for playlist prediction. In *Proceedings of 13th International Conference on Music Information Retrieval*, Porto, Portugal.

[O'Connor et al., 2002] O'Connor, M., Cosley, D., Konstan, J., and Riedl, J. (2002). Polylens: A recommender system for groups of users. In *ECSCW 2001*.

[Oliver and Kreger-Stickles, 2006] Oliver, N. and Kreger-Stickles, L. (2006). Papa: Physiology and purpose-aware automatic playlist generation. In *Proceedings of 7th International Conference on Music Information Retrieval*, Victoria, Canada.

[Pampalk and Gasser, 2006] Pampalk, E. and Gasser, M. (2006). An implementation of a simple playlist generator based on audio similarity measures and user feedback. In *Proceedings of 7th International Conference on Music Information Retrieval*, Victoria, Canada.

[Pampalk and Goto, 2007] Pampalk, E. and Goto, M. (2007). Musicsun: A new approach to artist recommendation. In *Proceedings of 8th International Conference on Music Information Retrieval*, Vienna, Austria.

[Pampalk et al., 2005] Pampalk, E., Pohle, T., and Widmer, G. (2005). Dynamic playlist generation based on skipping behavior. In *6th International Conference on Music Information Retrieval*, pages 634–637.

[Pauws and Eggen, 2002] Pauws, S. and Eggen, B. (2002). Pats: Realization and user evaluation of an automatic playlist generator. In *Proceedings of 3rd International Conference on Music Information Retrieval*, Paris, France.

[Pauws et al., 2006] Pauws, S., Verhaegh, W., and Vossen, M. (2006). Fast generation of optimal music playlists using local search. In *Proceedings of 7th International Conference on Music Information Retrieval*, Victoria, Canada.

[Pauws and Wijdeven, 2005] Pauws, S. and Wijdeven, S. v. d. (2005). User evaluation of a new interactive playlist generation concept. In *Proceedings of 6th International Conference on Music Information Retrieval*, London, UK.

[Ragno et al., 2005] Ragno, R., Burges, C., and Herley, C. (2005). Inferring similarity between music objects with application to playlist generation. In *7th ACM SIGMM International Workshop on Multimedia Information Retrieval*.

[Salvador and Chan, 2007] Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*.

[Schedl and Flexer, 2012] Schedl, M. and Flexer, A. (2012). Putting the user in the center of music information retrieval. In *Proceedings of 13th International Conference on Music Information Retrieval*, Porto, Portugal.

[Seyerlehner et al., 2009] Seyerlehner, K., Knees, P., Schnitzer, D., and Widmer, G. (2009). Browsing music recommendation networks. In *Proceedings of International Conference on Music Information Retrieval*, Kobe, Japan.

[Shao et al., 2009] Shao, B., Wang, D., Li, T., and Ogihara, M. (2009). Music recommendation based on acoustic features and user access patterns. *IEEE Transactions on Audio, Speech And Language Processing*, 17(8):1602–1611.

[Shardanand, 1994] Shardanand, U. (1994). Social information filtering for music recommendation. Master's thesis, Massachusetts Institute of Technology.

[Shardanand and Maes, 1995] Shardanand, U. and Maes, P. (1995). Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Denver, Colorado, USA.

[Su and Yeh, 2010] Su, J. and Yeh, H. (2010). Music recommendation using content and context information mining. *IEEE Intelligent Systems*, 25:16–26.

[Symeonidis et al., 2008] Symeonidis, P., Ruxanda, M. M., Nanopoulos, A., and Manolopoulos, Y. (2008). Ternary semantic analysis of social tags for personalized music recommendation. In *Proceedings of International Conference on Music Information Retrieval*, Philadelphia, Pennsylvania, USA.

[Tiemann and Pauws, 2007] Tiemann, M. and Pauws, S. (2007). Towards ensemble learning for hybrid music recommendation. In *Proceedings of 8th International Conference on Music Information Retrieval*, Vienna, Austria.

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

[Wu et al., 2010] Wu, Q., Burges, C., Svore, K., and Gao, J. (2010). Adapting boosting for information retrieval measures. *Journal of Information Retrieval*.

[Xu and Li, 2007] Xu, J. and Li, H. (2007). Adarank: a boosting algorithm for information retrieval. In *Proceedings of ACM SIGIR Conference Special Interest Group On Information Retrieval*.

[Yoshii and Goto, 2009] Yoshii, K. and Goto, M. (2009). Continuous pLSI and smoothing techniques for hybrid music recommendation. In *Proceedings of International Conference on Music Information Retrieval*, Kobe, Japan.

[Yoshii et al., 2006] Yoshii, K., Goto, M., Komatani, K., Ogata, T., and Okuno, H. (2006). Hybrid collaborative and content-based music recommendation using probabilistic model with latent user prefernces. In *7th International Conference on Music Information Retrieval*, pages 296–301.

[Yoshii et al., 2007] Yoshii, K., Goto, M., Komatani, K., Ogata, T., and Okuno, H. G. (2007). Improving efficiency and scalability of model-based music recommender system based on incremental training. In *Proceedings of 8th International Conference on Music Information Retrieval*, Vienna, Austria.