2012-05-07

A Music Recommendation System Based on User Behaviors and Genre Classification

Yajie Hu
University of Miami, huyajiecn@gmail.com

UNIVERSITY OF MIAMI

A MUSIC RECOMMENDATION SYSTEM BASED ON USER BEHAVIORS AND GENRE CLASSIFICATION

By

Yajie Hu

A THESIS

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science

Coral Gables, Florida

May 2012

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

A MUSIC RECOMMENDATION SYSTEM BASED ON USER BEHAVIORS AND GENRE CLASSIFICATION

Yajie Hu

Approved:

Mitsunori Ogihara, Ph.D. Professor of Computer Science	Terri A. Scandura, Ph.D. Dean of the Graduate School
Hüseyin Koçak, Ph.D. Associate Professor of Computer Science adfasdfasd	Burton Rosenberg, Ph.D. Associate Professor of Computer Science

HU, YAJIE (M.S., Computer Science)
(May 2012)

A Music Recommendation System
Based on User Behaviors And
Genre Classification

Abstract of a thesis at the University of Miami.

Thesis supervised by Professor Mitsunori Ogihara
Number of pages in text: (47)

This thesis presents a new approach to recommend suitable tracks from a collection of songs to the user. The goal of the system is to recommend songs that are preferred by the user, are fresh to the user's ear, and fit the user's listening pattern. "Forgetting Curve" is used to assess freshness of a song and the user log is used to evaluate the preference. I analyze user's listening pattern to estimate the level of interest of the user in the next song. Also, user behavior is treated on the song being played as feedback to adjust the recommendation strategy for the next one. Furthermore, this thesis proposes a method to classify songs in the Million Song Dataset according to song genre. Since songs have several data types, several sub-classifiers are trained 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. Finally, I develop an application to evaluate our approach in the real world.

TABLE OF CONTENTS

LIST OF TABLES v

LIST OF FIGURES vi

CHAPTER

1 Introduction 1

1.1 Motivations 1

1.2 Factors for music recommendation 2

1.3 Novel approaches 5

1.4 Organization 6

2 Background 8

2.1 RapidMiner 8

2.2 Datasets 10

2.2.1 Million Song Dataset 10

2.2.2 musiXmatch Dataset 13

2.2.3 Last.fm Dataset 13

3 Related Work 15

4 Proposed Method 20

4.1 Genre 22

4.1.1 Building up the genre similarity matrix 22

4.1.2 Genre prediction for the next song 23

4.1.3 Genre classification 26

4.2 Publish year 29

Page

4.3 Freshness . . . 29
4.4 Favor . . . 31
4.5 Time pattern . . . 33
4.6 Integrate into the final score . . . 33
4.7 Cold start . . . 34
5 Experiment . . . 36
5.1 Music recommendation system . . . 36
5.1.1 Data collection . . . 36
5.1.2 Results . . . 37
5.2 Song genre classification . . . 40
5.2.1 Experiment data . . . 40
5.2.2 Experiment results . . . 41
6 Conclusion . . . 45
REFERENCES . . . 46

LIST OF TABLES

Table		Page
1	55 fields provided in each per-song HDF5 file in the MSD.	12
2	Data sources	41
3	Experiment result comparison	44

LIST OF FIGURES

Figure		Page
1	The design prospective of the RapidMiner	9
2	Genius recommendation system in iTunes	16
3	Pandora recommendation system	16
4	Last.fm recommendation system	18
5	Genre Sample in AllMusic.com	23
6	Predict the next genre	25
7	Predict the next year	29
8	The Forgetting Curve	30
9	The appearance of NextOne Player	37
10	Running time of recommendation function	38
11	Representing the user logs to express favordness over a month	39
12	The distribution of continuous skips	40
13	Genre Samples in AllMusic.com	41
14	Confusion matrixes of four sub-classifiers	43
15	Confusion matrixes by all data	44

Chapter 1

Introduction

1.1 Motivations

As users accumulate digital music in their digital devices, the problem arises for them to manage the large number of tracks in their devices. If a device contains thousands of tracks, it is difficult, painful, and even impractical for a user to pick suitable tracks to listen to without using pre-determined organization such as playlists. The topic of this thesis is computationally generated recommendations. Music recommendation is significantly different from other types of recommendations, such as those for movies, books and electronics. Because a same song can be recommended to a same users many times if we successfully keep from him/her bored with it.

A main purpose of a music recommendation system is to minimize user's effort to provide feedback and simultaneously to maximize the user's satisfaction by playing appropriate song at the right time. Reducing the amount of feedback is an important point in designing recommendation systems, since users are in general lazy. We can evaluate user's attitude towards a song by examing whether the user listens to the song entirely, and if not, how large a fraction he/she does. In particular, we assume that if the user skips a recommended song, it is a bad recommendation, regardless of the reason behind it. If the recommended song is played completed, we infer that the user likes the song and it is a satisfying recommendation. On the other hand, if the song is skipped while just lasting a

few seconds, we conclude that the user dislikes the song at that time and the recommendation is less effective.

Using this idea we propose a method to automatically recommend music in a user's device as the next song to be played. In order to keep small the computation time for calculating recommendation, the method is based on user behavior and high-level features but not on content analysis. Which song should be played next can be determined based on various factors. In this paper, we use five factors: favor, freshness, time pattern genre and year.

1.2 Factors for music recommendation

Obviously, the favorite songs are supposed to have high priority in recommending music. Hence, favor is a significant factor to decide what song should be recommended.

However, if the favorite songs are recommended again and again in a short time, the user is bound to be bored. Freshness is thus introduced to the recommendation system. The system recommends fresh music to users. The freshness means that there is no record of playing the song to the user or the user has not been played it for a long time. The fresh music is more likely to attract user's attention and to give the user joyful experience.

Users have different tastes and preferences at different times. For instance, a user may prefer relaxing music in the afternoon and he/she may be keen on listening to exciting music in the evening. Similarly, the preference may change from weekdays to weekends. The time pattern therefore should be placed an

emphasis in the recommendation system in order to follow the variation of user taste according to the time pattern.

The difference from state-of-the-art recommendation methods is denying the assumption that user would like songs with similar genre. Some users prefer songs from a single genre while some others love songs from mixed genre. Hence, our recommendation system recognizes the change pattern of user taste according to song genre using time series analysis method. The genre of the next song is predicted by the change pattern instead of the similarity to the genre of the current song.

Most of song files record the genre in the header of files in the ID3v1 and ID3v2 formats. However, some songs have an invalid header. For example, the web site that provides the song would like to paste its URL as the genre tag in the file's header. In music recommendation, many methods see song genre as important metadata for retrieving songs. It is necessary to detect the invalid genre and complement it by automatically genre classification.

There is no genre dataset huge enough to cover mostly songs. However, other music dataset with various kinds of metadata and acoustic features are available. As the largest currently available dataset, the Million Song Dataset (MSD) is a collection of audio features and metadata for a million contemporary popular music tracks. The musiXmatch partners with MSD and provides a large collection of song lyrics in bag-of-word format. All of these lyrics are directly associated with MSD tracks. The Last.fm dataset is currently the largest

collection of song-level tags that can be used for research. We use these datasets to classify songs in terms of song genre.

Some papers have discussed the importance of using multiple data sources in genre classification and have proposed methods to use them. Most of these methods concatenated features from different data sources into a vector to represent a song [McKay et al., 2010]. However, for a very large dataset, it is impossible to ensure that every instance has valid data in all data sources. It is inevitable for the classification results to be reduced due to missing data influence in the concatenated vector.

If we have multiple classifiers and aggregate their assertions by voting, the accuracy of each classifier represents the authority of the "expert". Because the types of input data are different, the views of experts are not uniform. 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.

We extract features from audio, artist terms, lyrics and social tags to represent songs and train sub-classifiers. The trained sub-classifiers are combined to predict song genre. The songs with missing data in certain data types are classified using only available data.

The genre dataset is able to complement song genre when the genre tag of the song is invalid.

Similarly, the recommendation system also predicts the year of the next song using time series analysis method.

Finally, these five factors have dynamic weights to influence the recommendation results since a user has different emphasis on these factors in different time. We propose an algorithm to adjust the weights based on the user's feedback.

1.3 Novel approaches

In the recommendation system presented in this theses, several novel methods are proposed. These novel methods focus on music recommendation in the real world to adapt users' playing habit and meet the challenge of huge data.

1. *Breaking the assumption that the next song must be similar to the current song.* Instead of the assumption, this recommendation system predicts the next song's genre and publish year by time series analysis. This approach is better to accord with the change in the user's preference.

2. *Considering the time pattern of playing behaviors.* The time background of the playing behaviors is taken into consideration. In different time, users perhaps have different favorite music. The change partly depends on the time pattern of users' playing behaviors.

3. *Dynamic weights of factors to recommend the next song.* This thesis proposes a new approach to dynamically adjust the weights of five factors since users' taste is static. The weights of factors are able to converge to

the users' taste when the taste changes. The taste changes are realized by the user's feedback.

4. *Classifying song genre using sub-classifiers based on both sub-classifiers' authority and classification confidence.* In order to achieve a desired level of performance, we collect different types of song feature and train several sub-classifiers. The predictions of test samples by these sub-classifiers are integrated by sub-classifiers authority and confidence.

1.4 Organization

Chapter 2 introduces the tool and some datasets used in this thesis.

The major methods and applications of music recommendation are presented in Chapter 3. The methods and applications are categorized in different views. Each type of recommendation method has its own advantages and disadvantages and fit to some particular situations. Chapter 3 presents these methods and discusses their characteristics.

In Chapter 4, the proposed recommendation method and song genre classification approach are described. 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 proposed algorithm. Because the genre tag of a song file is sometimes invalid, a genre classification method automatically classifies songs in a huge dataset. The classification result is stored as a song-genre table in order to complement the genre data when the song file has no genre tag. This classification method applies

several sub-classifiers to deal with different types of the data source and then calculates the final classification result from the results of these sub-classifiers.

We evaluate the recommendation method and song genre classifier performance in Chapter 5. A recommendation system is implemented and used by volunteers. The evaluation result of the recommendation method is satisfied. We build a collection of songs with genre tags from AllMusic.com as the ground truth. The genre classification result in this ground-truth data surpasses the baselines and is competitive to the results in similar tasks.

Chapter 6 summarizes the recommendation method and the song genre classification method.

Chapter 2

Background

This chapter introduces the tool and datasets that are used in this thesis.

2.1 RapidMiner

This thesis uses RapidMiner to test several classification methods and classify songs according to song genre.

RapidMiner provides data mining and machine learning procedures including: data loading and transformation (ETL), data preprocessing and visualization, modeling, evaluation, and deployment [RapidMiner, 2012]. The data mining processes can be made up of arbitrarily nestable operators, described in XML files and created in RapidMiner's graphical user interface (GUI). RapidMiner is written in the Java programming language. It also integrates learning schemes and attributes evaluators of the Weka machine learning environment [Weka, 2012] and statistical modeling schemes of the R-Project.

Available functionalities include:

- Bypassing its data mining functions and generating its own figures.
- Exploring data in the Microsoft Excel format ("knowledge discovery").
- Constructing custom data analysis workflows.
- Calling RapidMiner functions from programs written in other languages/systems (e.g. Perl).

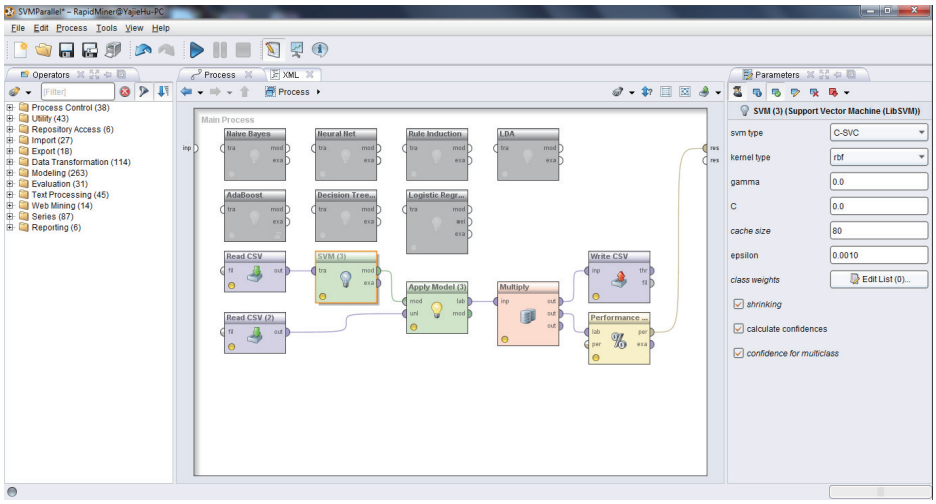


Figure 1: The design prospective of the RapidMiner

Features:

- Broad collection of data mining algorithms such as decision trees and self-organization maps.
- Overlapping histograms, tree charts and 3D scatter plots.
- Many varied plugins, such as a text plugin for doing text analysis.

RapidMiner provides major of classification methods and the parameters of these methods are able to be edited. It is very convenient do classification experiments and test different classification methods. What the user needs to do is to replace the corresponding module of the classifier and run the system again. The modeling design makes the process quite clear, understandable and flexible as shown in Figure 1.

In the Figure 1, the grey modules are other candidate classifiers and we can test these classifiers.

2.2 Datasets

In this thsis, we need to cover most of songs and label genre tags for them. If a song file doesn't have genre tags, the system will retrieve the song's genre from the song-genre table. There does not exist publicly accessible large dataset with song genre, but there are very large datasets with other types of data. The song genre could be recognized from these types of data. The datasets that will be used in Chapter 4 are listed below.

2.2.1 Million Song Dataset

The Million Song Dataset is a freely-available collection of audio features and metadata for a million contemporary popular music tracks.

Its purposes are:

- To encourage research on algorithms that scale to commercial sizes
- To provide a reference dataset for evaluating research
- To provide a shortcut alternative to creating a large dataset with APIs (e.g. the Echo Nest APIs)
- To help new researchers get started in the MIR field

The core of the dataset is the feature analysis and metadata for one million songs, provided by a company, The Echo Nest. The MSD contains audio features and metadata for a million contemporary popular music tracks. It contains:

- 280GB of data
- 1,000,000 songs/files
- 44,745 unique artists
- 7,643 unique terms (Echo Nest tags)
- 2,321 unique musicbrainz tags
- 43,943 artists with at least one term
- 2,201,916 asymmetric similarity relationships
- 515,576 dated tracks starting from 1922

Each song is described by a single file, whose contents are listed in Table 1 [Bertin-Mahieux et al., 2011].

The acoustic features related to song genre are extracted, such as bar starts, bar confidences, beats confidences, section starts, section confidences, segment loudness max, segment pitches, segment timbres and tempo. Each of them is a series of real values to represent the variance of the song in terms of certain kind of feature. The sequences of these features cannot be directly used in a vector to represent the song in a classifier. Therefore, we use the statistical measures of the sequences instead of the sequences to generate the vector, such as the mean, the variance, the Q values. $Q(0)$ is the minimum value of the sequence. $Q(1)$ is the one quarter quality factor of the sequence. $Q(2)$ is the intermediate quality factor of the sequence. $Q(3)$ is the three quarters quality

Table 1: 55 fields provided in each per-song HDF5 file in the MSD.

analysis_sample_rate	artist_7digitalid	artist_familiarity
artist_hotttnesss	artist_id	artist_latitude
artist_location	artist_longitude	artist_mbid
artist_mbtags	artist_mbtags_count	artist_name
artist_playmeid	artist_terms	artist_terms_freq
artist_terms_weight	audio_md5	bars_confidence
bars_start	beats_confidence	beats_start
danceability	duration	end_of_fade_in
energy	key	key_confidence
loudness	mode	mode_confidence
num_songs	release	release_7digitalid
sections_confidence	sections_start	segments_confidence
segments_loudness_max	segments_loudness_max_time	segments_loudness_start
segments_pitches	segments_start	segments_timbre
similar_artists	song_hotttnesss	song_id
start_of_fade_out	tatums_confidence	tatums_start
tempo	time_signature	time_signature_confidence
title	track_7digitalid	track_id
year		

factor of the sequence and $Q(4)$ is the maximum value of the sequence. The vector that consists of statistical measures has 46 real values. Most of values in a vector are non-zero.

The artist terms are extracted because artist terms describe the style of the artist and are related to the song genre. After cleaning and stemming, the user terms represent the artist in the bag-of-words format. The feature is binary and set to 1 if the term corresponding to the feature appears in the artist terms. The length of the artist terms vector is the number of total terms and reaches 1011. Most of the features are zero and a vector has average 25.74 non-zero features. The vector is very sparse.

2.2.2 musiXmatch Dataset

The musiXmatch dataset brings a large collection of song lyrics in bag-of-words format [musiXmatch, 2012]. All of these lyrics are directly associated with MSD tracks. The musiXmatch is able to resolve over 77% of the MSD tracks and releasing lyrics for 237,662 tracks. The other tracks were omitted for various reasons, including:

- Diverse restrictions, including copyrights
- Instrumental tracks
- The numerous MSD duplicates were skipped as much as possible

Since the lyrics describe the semantic content of the song, the content has the indirect relationship to the song genre. For example, the lyrics content of a rap song could be different from the lyrics content of a country song. Each track is described as the word-counts for a dictionary of the top 5,000 words across the set. The 5,000 words in the dataset account for 50,607,582 occurrences and there are 237,662 tracks. A track hence has average 212.94 words but the vector has 5,000 features.

2.2.3 Last.fm Dataset

The Last.fm Dataset brings the largest research collection of song-level tags and pre-computed song-level similarity [Last.fm, 2012]. All the data is associated with MSD tracks. Selected features of the Last.fm dataset are as follows:

- 943,347 matched tracks MSD and Last.fm
- 505,216 tracks with at least one tag
- 584,897 tracks with at least one similar track
- 522,366 unique tags
- 8,598,630 (track - tag) pairs
- 56,506,688 (track - similar track) pairs

Although tracks have many noisy tags, some tags related to song genre are able to explicitly point out the genre of the song. The social tags of the Last.fm dataset are therefore used to classify songs according to song genre in this thesis.

Chapter 3

Related Work

Various music recommendation approaches have been developed. We can categorize these approaches in several classes.

- *Automatical playlist generation* focuses on recommending songs that are similar to chosen seeds to generate a new playlist. Ragno et al. [Ragno et al., 2005] provided 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. They have an underlying problem that all seed-based approaches produce excessively uniform lists of songs if the dataset contains lots of music cliques. In iTunes, Genius employs similar methods to generate a playlist from a seed as shown in Figure 2.

- *Dynamic music recommendation* improves automatic play-list 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 3.

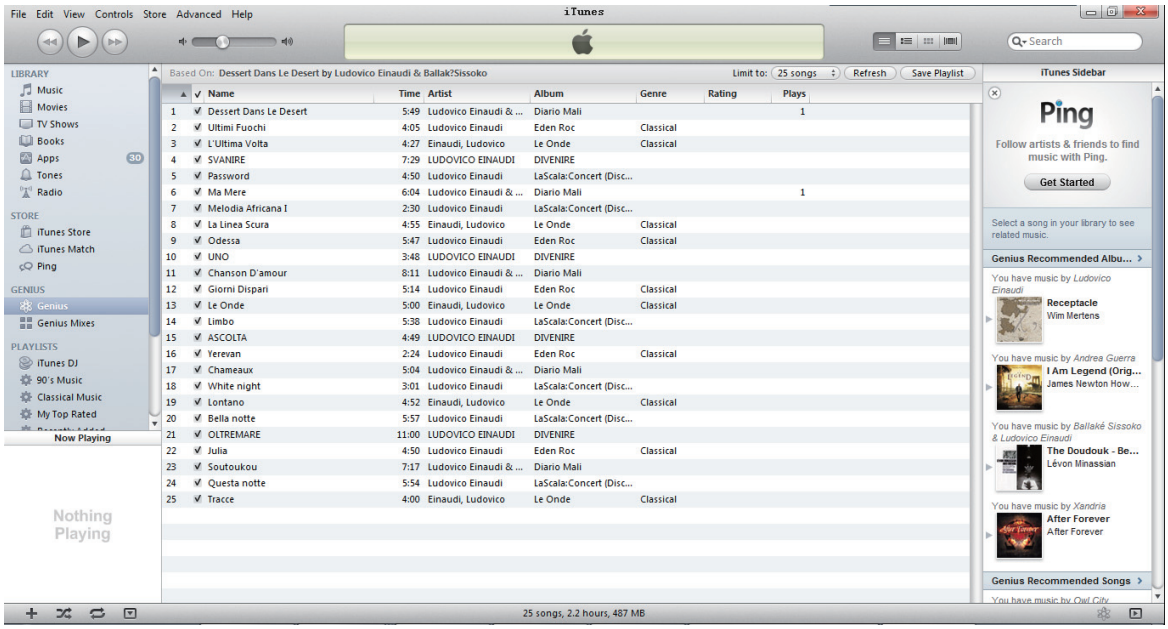


Figure 2: Genius recommendation system in iTunes

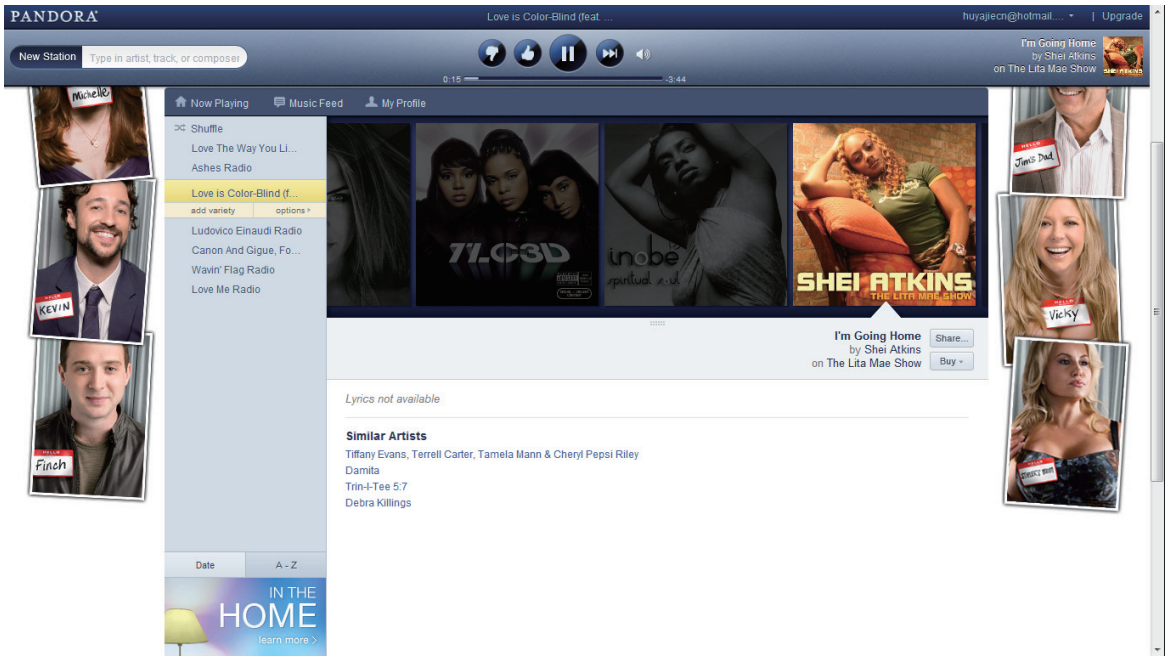


Figure 3: Pandora recommendation system

- *Collaborative-filtering methods* recommend pieces of music to a user based on rating of those pieces by other users with similar taste [Cohen and Fan, 2000]. However, collaborative filtering methods require many users and many ratings and are unable to recommend songs that have no ratings. Hence, users have to be well represented in terms of their taste if they need effective recommendation. This principle has been used by various social websites, including Last.fm (Figure 4), myStrands.

- *Content-based methods* compute similarity between songs, recommend songs similar to the favorite songs, and remove songs that are similar to the skipped songs. In an approach proposed by Cano et al. [Cano et al., 2005], acoustic features of songs are extracted, such as timbre, tempo, meter and rhythm patterns. Furthermore, some work expresses similarity according to songs emotion. Cai et al. [Cai et al., 2007] recommends music based only on emotion.

- *Hybrid approaches*, which combine music content and other information, are receiving more attention lately. Donaldson [Donaldson, 2007] 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.

- *Context-based methods* take context into consideration. Liu et al. [Liu et al., 2009] take the change in the interests of users over time into

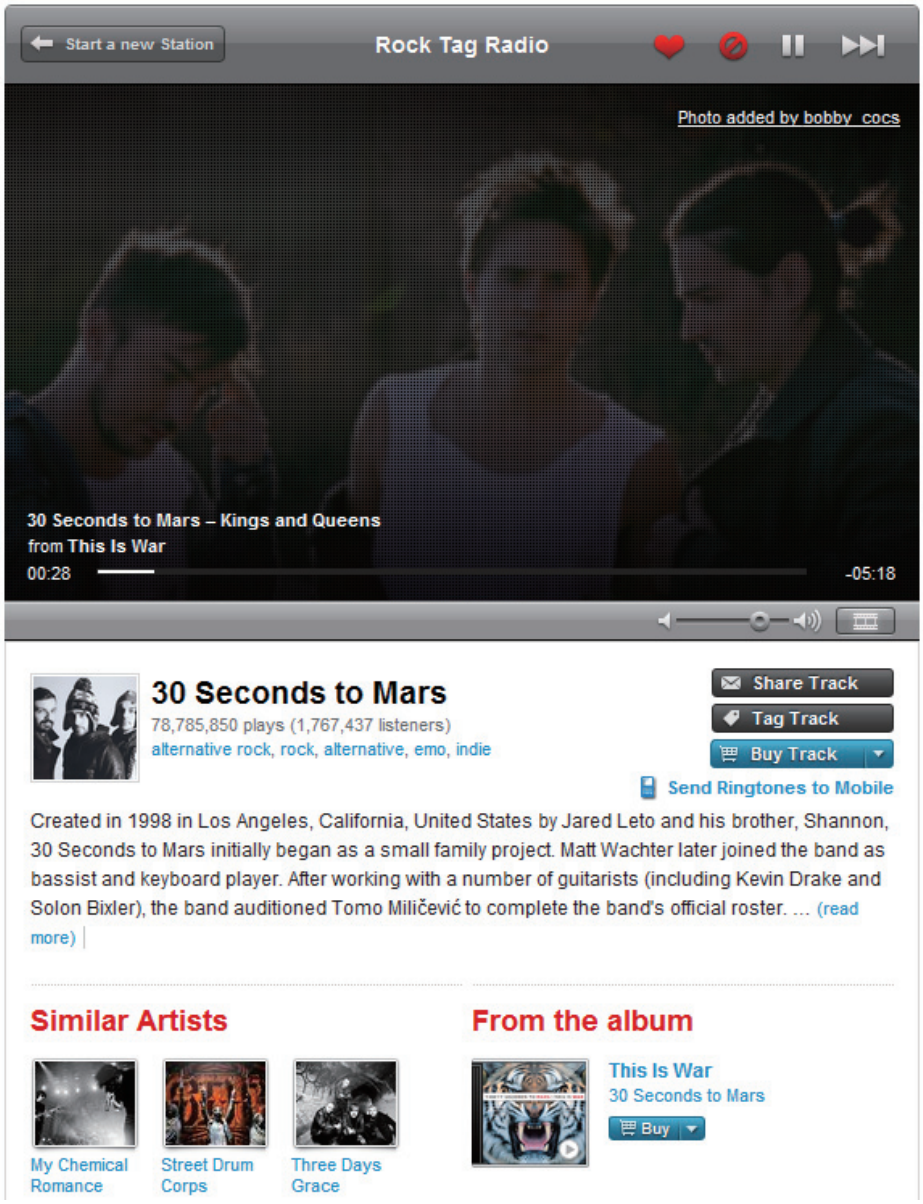


Figure 4: Last.fm recommendation system

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 using content analysis assists system to select appropriate songs.

The music recommendation of this thesis belongs to dynamic music recommendation and is similar to Pandora in terms of the way pieces are reccommended. However, the factors that are taken into consideration are different from state-of-the-art methods.

Chapter 4

Proposed Method

We determine whether a song is to be recommended as the next one in the playlist from five perspectives genre, year, favor, freshness and time pattern.

From genre and year perspectives, we use time series analysis to predict the genre and year of the next song rather than selecting the song with similar genre and year to the current song. The reason is that some users like listening similar songs according to genre and year while others perhaps love mixing songs and the variance on genre and year. Also, one user may have different preferences. We does not assume that a similar song to the current one can be reasonably seen as a good choice for recommendation. Prediction using time series analysis method caters to a user's taste better than the assumption.

Song genre is available in the header of MP3 file, like ID3v1 or ID3v2 tags. However, some songs have an empty header or their genre tags are invalid. For instance, the genre tag is some advertisements or other irrelevant content. If the recommendation system analyzes the acoustic features of the song, the computation complexity would make the system impractical. Users cannot wait for a recommendation result over several seconds, even though the recommendation result is just one the user love. Hence, the song genre is supposed to be pre-computed and stored in a table. The system is then able to retrieve the genre of a song from the table if the song has no valid genre tag.

In order to cover most of songs, the system needs a huge genre dataset but so far the dataset is unavailable. The system has to collect other large datasets and use them to classify the songs according to song genre. A song has several types of features, such as acoustic features, lyrics, social tags, artist information and so forth. Obviously, the more useful information is considered into the classification, the higher performance could be reached. As a result, it is necessary to propose an approach to integrate these types of feature.

Obviously, the system should recommend users' favorite songs to them. The amount of times of actively playing a song and the amount of times of completely listening a song can infer the strength of favor to the song. We collected user's behavior to analyze the favor of songs and the playing behavior is seen as the feedback to the song. The partition of playing the song is considered as the score of the song.

In a common sense, a few users like listening to a song again and again in a short time, even though the song could be the user's favorite. On the other hand, songs that used to be popular, like *Wavin' Flag*, *Waka Waka*, and that a user loved to listen to may be now old and a little bit insipid. However, if the system recommends them at a right time, the user may feel it is fresh and enjoy the experience. Consequently, we take freshness of songs into consideration.

Due to the work time and biological clock, users have different tastes in choosing music. In a different period of a day or a week, users tend to select different styles of songs. For example, in afternoon, a user may like a soothing

kind of music for relaxation and switch to energetic songs in evening. In this thesis, we use Gaussian Mixture Model to represent the time pattern of listening and compute the probability of playing a song at that time.

Finally, these factors should be integrated and the system should use the integrated score of these factors to determine which song should be the next song.

4.1 Genre

Recent playing sequence of a user represents the user's habit of listening so I analyze the playing sequence using a time series analysis method to predict the genre of the next song. The system records 16 recent songs that were played for duration over to their half-time mark. Since the ID3v1 or ID3v2 tags, are noisy, we developed a web wrapper to collect genre information from `AllMusic.com`, a popular music information website, and use that information to retrieve songs' genres. ID3v1 or ID3v2 tags will be used unless `AllMusic.com` has no information about the song. If both are not available, the system will retrieve the song's genre from the song-genre table.

4.1.1 Building up the genre similarity matrix

Furthermore, `AllMusic.com` not only has a hierarchical taxonomy on genre but also provides subgenres with related genres. The hierarchical taxonomy and related genres are shown in Figure 5.

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

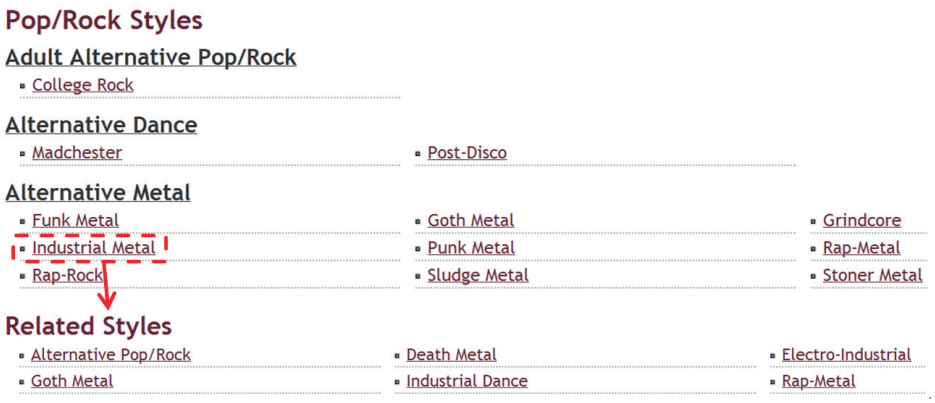


Figure 5: Genre Sample in AllMusic.com

An edge's value is set to 1.0, if two genres are connected by the edge are related. The parent relationship is valued at a different distance, which varies by the depth in the taxonomy, that is, high level corresponds to larger distance while low level corresponds to smaller distance. We thus assume the distance is transitive and update the distance graph as follows until there is no cell update.

$$E_{ij} = \min_k \left(E_{ij}, E_{ik} + E_{kj} \right), \tag{1}$$

where E_{ij} is the value of edge ij. Therefore, we obtain the similarity between any two kinds of genre and the maximum value in the matrix is 6.

4.1.2 Genre prediction for the next song

In this part, we try to predict the possible genre of the next song to fit the user's pattern rather than assuming the next genre is similar.

Now, the system converts the series of genres of recent songs into a series of similarity between neighbor genres using the similarity matrix. The series of similarity will be seen as the input for time series analysis method and we can

estimate the next similarity. Then, the current genre and the estimated similarity will give us genre candidates.

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

$$\Phi(B)(1-B)^d y_t = \delta + \Theta(B)\varepsilon_t \tag{2}$$

$$\Phi(B) = 1 - \sum_{i=1}^{p} \phi_i B^i \tag{3}$$

$$\Theta(B) = 1 + \sum_{i=1}^{q} \theta_i B^i, \tag{4}$$

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. 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 $t + \tau$. As an illustration consider forecasting the ARIMA$(1, 1, 1)$ process

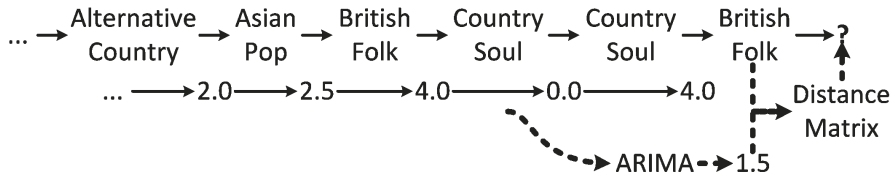


Figure 6: Predict the next genre

$$(1 - \phi B)(1 - B)\, y_{t+\tau} = (1 - \theta B)\, \varepsilon_{t+\tau} \tag{5}$$

$$\hat{\varepsilon}_t = y_t - \left[\delta + \sum_{i=1}^{p+d} \phi_i y_{t-i} - \sum_{i=1}^{q} \theta_i \hat{\varepsilon}_{t-i} \right] \tag{6}$$

Considering the benefit of ARIMA, the system employs it to fit the series of similarity and to predict the next similarity. The process is shown in Figure 6.

We use Gaussian distribution to evaluate the probability of the next genre as the score for the genre candidates. The genre, whose distance to the current genre is equal to the estimated distance, has the biggest probability.

$$p(g_t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(s(g_t, g_{t-1}) - \hat{\varepsilon}_t)}{2\sigma^2}}, \tag{7}$$

where $p(g_t)$ is the possibility that the next song's genre is g_t. $s(g_t, g_{t-1})$ describes the similarity between the genre g_t and the genre g_{t-1}. It is obtained from the genre similarity matrix built by the genre taxonomy of AllMusic.com. $\hat{\varepsilon}_t$ is the predicted similarity estimated by ARIMA.

4.1.3 Genre classification

Data types in genre classification

In order to cover most songs, it is necessary to build up a huge song-genre table. Hence, we need huge datasets to guarantee the table is practical and useful in this recommendation system. We used the several datasets introduced in Chapter 2.

Genre classification by sub-classifiers

Each type of features has individual characteristics so we apply each data source to respectively train a sub-classifier. It is possible to choose a particular classification method to train the sub-classifier for each data source. The classification method adapts to the type of features, like high sparsity or low dimensions.

A song has much possible genre so the classifier must determine the song to assign into a class among multiple classes. In order to reduce the classification complexity, the multi-class classification problem is reduce to a series of two-class classification problems, like Pop/Non-Pop, Blues/Non-Blues, Jazz/Non-Jazz, and so on. Then, the classification confidence for a particular class is used to determine which class the song belongs to. The class whose classification confidence is the highest one among these binary classification results is seen as the final classification result.

The main issue here is how to integrate the results predicted by the sub-classifiers into a final result.

Some voting methods use the authority of sub-classifiers to integrate results. The authority of a sub-classifier is estimated by a validation test. The sub-classifiers that have higher performance in the validation test are given higher authority values. The results are weighted by the authority of the corresponding sub-classifier. The integrated result is voted by these weighted results.

If we look into the voting methods, they are based on a subtle assumption that a particular sub-classifier has stable classification performance for every test sample. Hence, for any sample, the results have static weights. However, the fact is not as simple as the assumption shows.

For example, a sub-classifier trained by social tags classifies a sample with a genre tag, like "Rock". Even though the sub-classier doesn't have a high authority, the sub-classifier absolutely ensures that the song genre of this sample is "Rock". In other word, the sub-classifier has a full confidence to determine a particular sample into a class and so it must play a crucial role in this voting for the sample.

Based on this idea, this thesis proposes a method to integrate results based on both the sub-classifier authority and the classification confidence.

Let C be a classifier set that contains some n sub-classifiers, namely, $C = \{c_1, c_2, \ldots, c_n\}$. Suppose that songs are distributed into some m genres, $G = \{g_1, g_2, \ldots, g_m\}$. The voting result is shown in Equation 8 below.

$$\mathrm{G}\left(I_k\right) = \arg\max_{g_j} \left\{ \sum_{i=1}^{|C|} \left[\mathrm{Auth}\left(c_i\right) \cdot \mathrm{Conf}\left(c_i, g_j, I_k\right) \right] \right\} \tag{8}$$

Auth (c_i) denotes the authority of the classifier c_i and varies between 0.0 and 1.0. Auth (c_i) is estimated by the accuracy of the classification in the validation test.

Conf (c_i, g_j, I_k) is the confidence of the classifier c_i to classify the instance I_k to genre g_j. The confidence value is in the interval $[0.0, 1.0]$, where 1.0 means the classifier has no doubt to classify a sample into a class and 0.0 means the classifier denies assigning the sample into the class. 0.5 shows the classifier is not sure to make a decision. Note that the sum of the confidence for the two classes of a binary classifier, is always 1.0. Different classification methods have different measures to estimate the classification confidence. The following list discusses the measures for the classification methods that are employed in this thesis.

- *Naïve Bayes.* For Naïve Bayes, the posterior probability is seen as the confidence for a class.

- *Neural Net.* Neural Net has normalized real value output from -1.0 to 1.0. A positive value means the confidence to assign the instance to a positive label.

- *Logistic Regression.* We employ the approach proposed by Lee [Lee, 2010] to estimate the confidence for logistic regression.

- *Support Vector Machines.* The margin from the instance location to the classification hyper plane is considered to be the confidence of the SVM classifier.

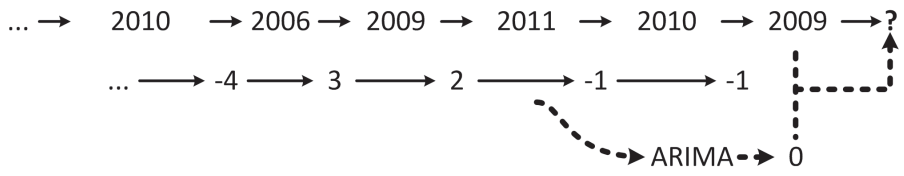


Figure 7: Predict the next year

The confidence values 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 Publish year

The publish year is similar to genre so we use ARIMA to predict the next possible publish year and compute the probability of a publish year. Figure 7 shows the prediction process.

4.3 Freshness

As a new approach of this thesis, we take freshness of a song for a user into consideration. Many recommendation systems, such as the one [Logan, 2004] is based on metadata of music, do not keep record of what pieces are recommended before or user response, and many repeatedly recommend the same music over and over again. Furthermore, if the system keeps track of the count of plays while ignoring user feed back, songs that are recommended over and over again may be recognized as favorite songs. The iteration makes users fall into a "favorite trap" and feel bored. Therefore, an intelligent recommendation system should avoid recommending a same set of songs many times in a short period.

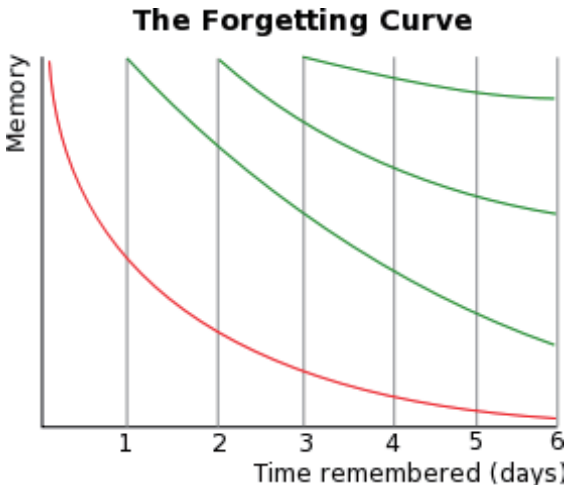


Figure 8: The Forgetting Curve

On the other hand, the system is supposed to recommend some songs that have not been played for a long time because these songs are fresh for users even though they once listened to them multiple times.

Freshness can be considered as the strength of strangeness or the amount of forgetting part in mind. Hence, we apply the Forgetting Curve [Ebbinghaus, 1913] to evaluate the freshness of a song for a user. The Forgetting Curve is calculated by Equation 9.

$$R = e^{-\frac{t}{S}}, \tag{9}$$

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

The Forgetting Curve is plotted as shown in Figure 8. Theses curves show the memory fade out in different strength of memory.

Lesser the amount of memory retention of a song in a user's mind is, more fresh the song is for the user. In our work, S is defined as playing times and t is the period from the last time of playing the song till current. 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.

4.4 Favor

The strength of favor for a song plays a rather important role in recommendation. In playing songs, the system should give 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 tends to listen to a favorite song more frequently than the others and thus he/she listens to a large portion that the others, if he/she does not listen to it entirely.

In this thesis, we see the feedbacks as rating behaviors. If the user listens a song completely, the rating to the song is positive and set to 1.0. If the user skips the song at the beginning of the song, the behavior implies the rating is 0.0. The rating score depends on the amount of the partition of the song played and the region is $[0.0, 1.0]$.

The average score or the sum score is not a reasonable approach to estimate the song's favor to a user. Let simplify the score to 0.0 or 1.0 to analyze the rating approach. For instance, a song A has been played 50 times and has 40 positive scores, namely 1.0, and 10 times negative scores, namely 0.0. A song B

has been played 5 times and all of these scores are 1.0. Which song is more favorite one? The average score of B is higher than that of A. However, the sum of the scores of A is further more than that of B. The great number of positive scores make the system have strong confidence to conclude that A is a favorite. On the other hand, the small number of playing B cannot solidly support the conclusion that the user prefers B to A.

We refers to the approach applied by the Internet Movie Database (IMDb) [IMDB, 2012], an online database of information related to movies, television shows, actors and so on.

The approach is based on the Bayesian probability on user ratings. The rating of a movie is calculated by a true Bayesian estimate:

$$WR = \frac{v}{v+m}R + \frac{m}{v+m}C, \tag{10}$$

where R is the average rating for the movie, v denotes the number of votes for the movie. m is the minimum votes required to be listed in the Top 250 (currently 3000) and C is equal to the mean vote across the whole report (currently 6.9). WR is the weighted rating of the song.

In this thesis, R is set to the mean partition of songs playing, v the number of playing for the song, m the minimum number of playing required to be listed in the top 20% songs, C and the mean partition of song playing across the whole songs.

This approach help avoid a situation in which a song with a few playing is always rated a low score or radical fluctuations. Songs are expected to be rated an almost equal much of times, hence, the rating is added a mean score C with a minimum number of the ratings in the top 20% songs. When the song has a very few ratings, the weighted rating is close to the mean score C. When the song has plenty of ratings, the weighted rating is approximately equal to the rating of the score R.

4.5 Time pattern

Since users have different habits or tastes in different period of a day or a week, our recommendation system takes time pattern into consideration based on user log. The system records the time of the day and week those songs are played. Then, Gaussian Mixture Model is employed to estimate the probability of playing at a specific time. The playing history of a song in different periods trains the model using Expectation Maximization algorithm. When the system recommends songs, the model is used to estimate the probability of the song being played at that time.

4.6 Integrate into the final score

A song is assessed whether it is a fit for recommendation as the next song from the aforementioned five perspectives. In order to rank results and select a song as the next song, 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 in integration. We

calculate these weights using Gradient Descent so at to the system recommendation close to the user's needs. However, it is silly to offer many possible recommendation results and determine how to descent based on user's interaction. We use the recent recommendation results to adjust the weights, which is initialized by $(1.0, 1.0, 1.0, 1.0, 1.0)$, as shown in Algorithm 1.

4.7 Cold start

Cold start is an important problem for building recommendation systems. At the beginning, the system has no idea 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 et al.'s work [Pampalk et al., 2005]. After 16 songs, the system uses the metadata of these songs and user behavior to recommend a song as the next one.

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 χ_i, which is `like` or `dislike`, and the score of each factor of the first recommendation, $\mathbf{\Lambda}_i$, and that of the second one, $\mathbf{\Lambda'}_i$.
Descent step Δ, which is positive.
Current factor weights, $\mathbf{W}$.
Output: New factor weights, $\mathbf{W'}$.
Process:
if χ_t = ***dislike*** **then**
  Initialize an array $\mathbf{F}$ to record the contribution of each factor.
  for R_{t-k+1} **to** R_t **do**
    $\Delta\mathbf{\Lambda}_i = \mathbf{\Lambda}_i - \mathbf{\Lambda'}_i$
    $max = \arg\max_j \left(\Delta\lambda_j\right)$
    $min = \arg\min_j \left(\Delta\lambda_j\right)$
    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
  $inIndex = \arg\max_i \left(\mathbf{F}\right)$
  $w'_{inIndex} = w_{inIndex} + \Delta$
  $w'_{i,i\neq inIndex} = w_i - \Delta/(dimension - 1)$
  $deIndex = \arg\min_i \left(\mathbf{F}\right)$
  $w'_{deIndex} = w'_{deIndex} - \Delta$
  $w'_{i,i\neq deIndex} = w'_i + \Delta/(dimension - 1)$
end
else
  $\mathbf{W'} = \mathbf{W}$
end
return $\mathbf{W'}$

Chapter 5

Experiment

This part presents the performance of the genre classification method comparing to some baselines methods.

5.1 Music recommendation system

5.1.1 Data collection

An application system, called NextOne Player[1], is developed to collect run-time data and user behavior for this experiment. 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 provides recommendation function using the approach described in Chapter 4 and also collects data for performance evaluation. The recommendation will work when the current song in the playlist ends or `NextOne` button is clicked. The appearance of the application is shown in Figure 9. The `Like it` and `Dislike it` buttons are used to collect user feedback. The proportion of a song played is recorded and viewed as the measure of satisfaction of a user for the song.

In order to compare our method with random selection, the player selects one of the two methods when it is loaded. The probability of running each method is 0.5. Everything is exactly same except the recommendation method. In the contrasting experiment, users cannot realize which method is selected.

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

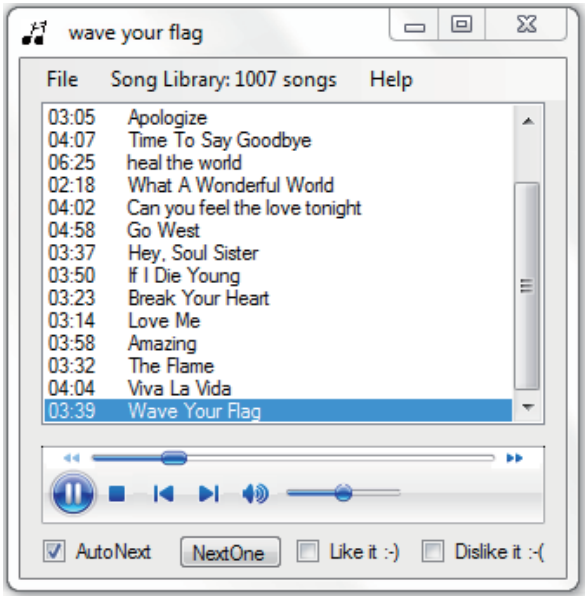


Figure 9: The appearance of NextOne Player

We have collected data from 11 volunteers. They consist of 9 graduate students and 2 professors and include 3 female students. They use the application in their devices which recommend songs from their own collections so the experiment is run on open datasets.

5.1.2 Results

First, we show the running time of recommendation function as it is known to have a major influence on the user experience. The running time results appear to be in an acceptable range. We run the recommendation system for different magnitudes of the song library and at each size the system recommends 32 times[2]. Figure 10 shows the variation in running time with the corresponding variations to the size of song library. We observe that the running time increases linearly with the increase in size of the song library. In order to

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

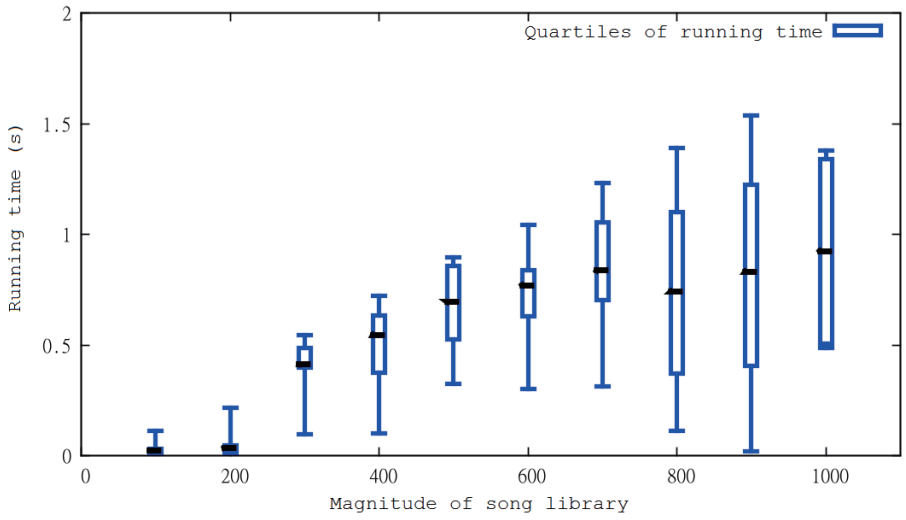


Figure 10: Running time of recommendation function

provide a user-friendly experience, the recommendation results are processed near the end of the current song that is playing, and the result is generated when the next song begins.

From Figure 10, it is reasonable to conclude that the system has an acceptable running time in personal devices since the scale of the song data is not too large.

In order to evaluate the approach, the system records the playing behavior of the user. We collected the user logs from volunteers and calculated the average proportion of playing song length, which means how much partition of a song is played before it is skipped. Under the assumption that the partition implies the "favoredness" of the song for a user, we evaluate the recommendation approach by the partition as shown in Figure 11, where the histograms represent the number of songs that were played on a day. The curves in the graph represent the variation of the "playing proportion". The range of these two curves is

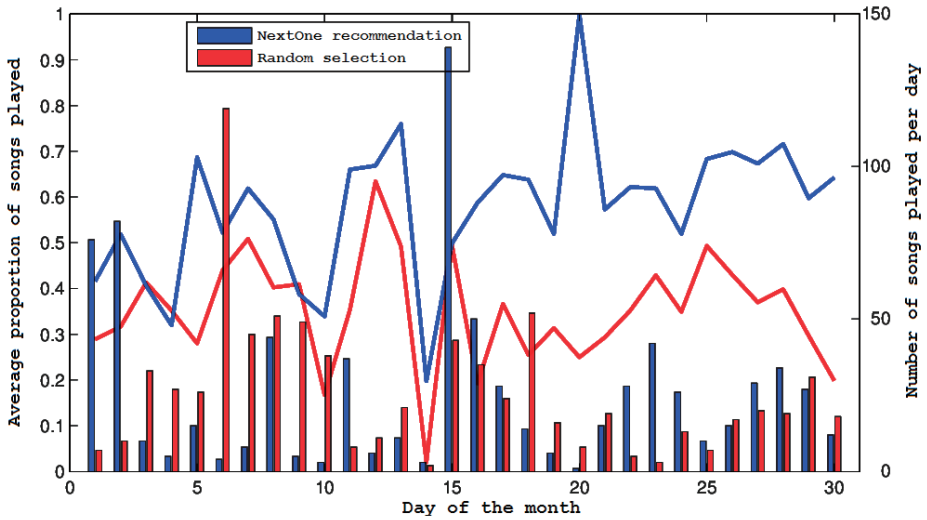


Figure 11: Representing the user logs to express favordness over a month

$[0.0, 1.0]$ and 1.0 is the best performance of the experiments.

In Figure 11, the histograms represent the number of songs that were played on a day. The curves in the graph represent the variation of the "playing partition".

Let us define a skip be changing to the next track by the user before playing 5% of the length of the current track. If a recommendation system cannot recommend proper songs so many times that the user skips songs again and again, the system will lose the user's interest. Continuous skips therefore have a significant negative influence on the user experience. It is almost inevitable for a recommendation system to mismatch the user's current taste but the capability to adjust the recommendation strategy quickly represents the robustness and intelligence of the system. An intelligent recommendation system is supposed to cater to the user's taste in a few unsatisfied recommendations. We use the number of continuous skips to measure the robustness and intelligence of

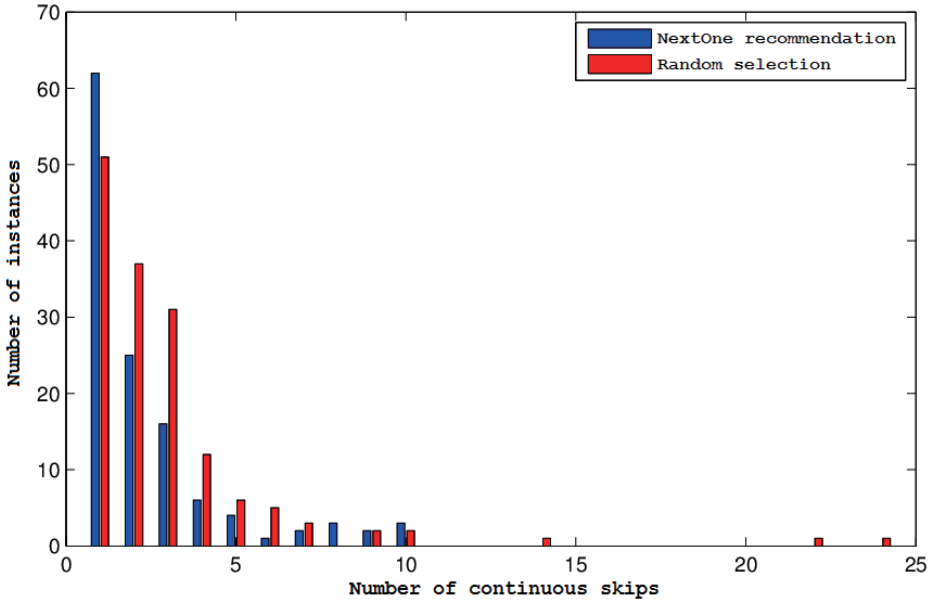


Figure 12: The distribution of continuous skips

the recommendation system. Figure 12 shows the distribution of continuous skips using our method and random selection.

From Figures 11 and 12, we can conclude that the recommendation approach surpasses the baseline and our recommendation is effective. Our approach is able to fit to a user's taste, and adjust the recommendation strategy quickly whenever user skips a song.

5.2 Song genre classification

5.2.1 Experiment data

In our experiment, we applied MSD, MusiXmatch and Last.fm tag datasets to extract features, as shown in Table 2. The records in these data sources are matched via `trackID`.

Table 2: Data sources

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

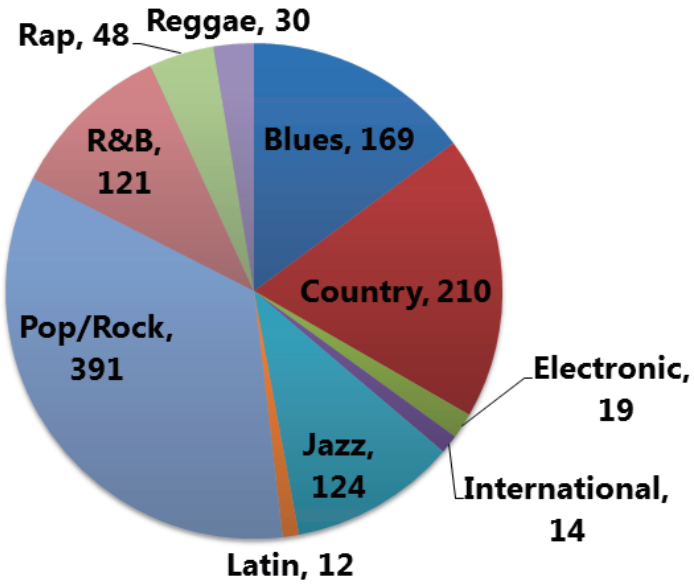


Figure 13: Genre Samples in AllMusic.com

AllMusic.com provides genre taxonomy, which consists of 10 major genres with sample songs. Some music or radio service websites organize songs by similar genre classes. Thus, this song genre taxonomy is rational and practical and this thesis classifies songs according to this genre taxonomy. 1,138 songs are collected from AllMusic.com and they have valid records in MSD as the ground truth. The distribution of the songs according to genre is shown in Figure 13.

5.2.2 Experiment results

In order to improve classification performance, we convert multi-class classification into a series of binary classifications. Thus, the classification result

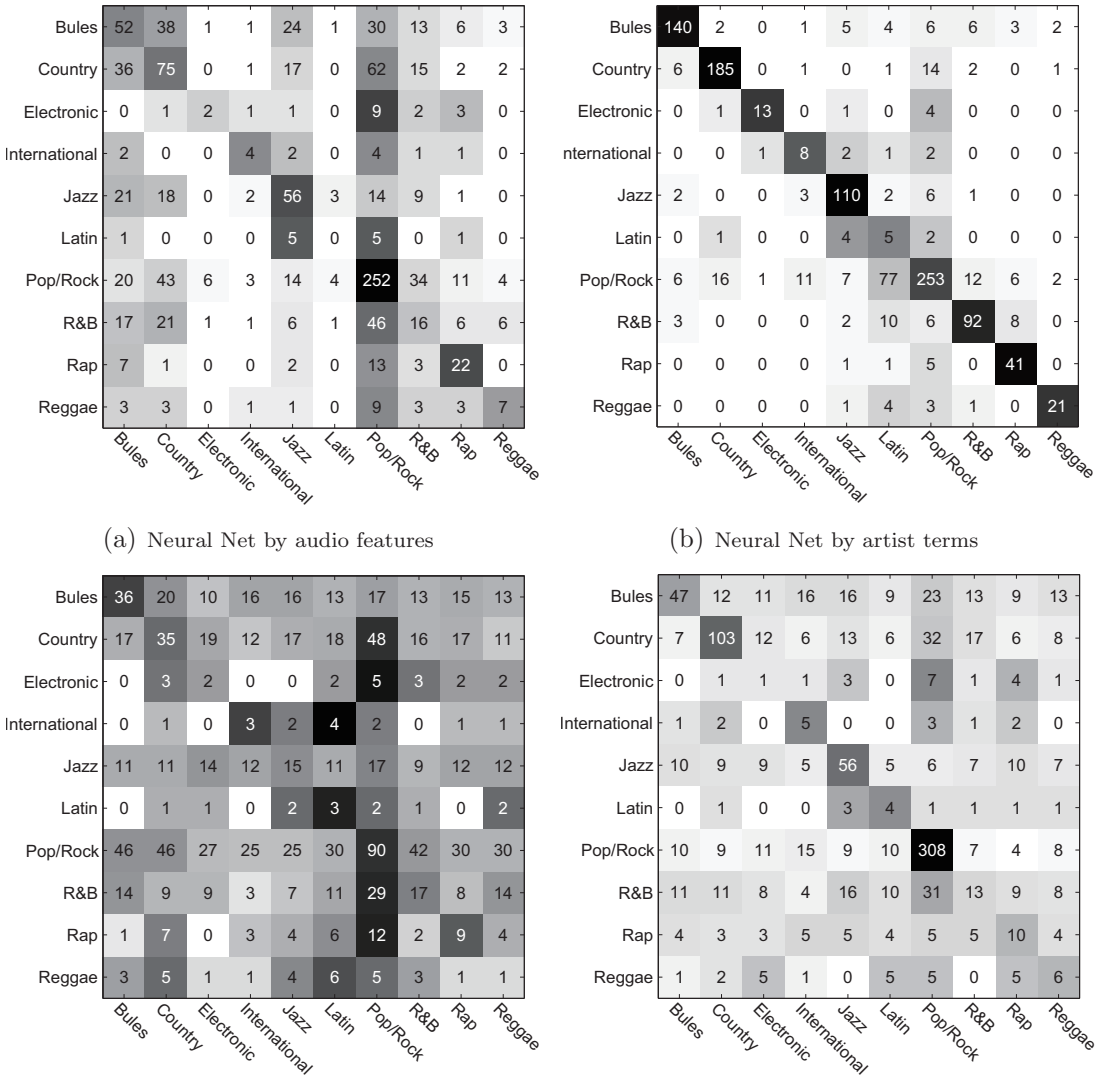
of a song is a vector of confidence to classify the song into a particular genre. The predicted genre is the one whose confidence is highest.

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 performance is evaluated by 5-folder cross validation. The best performance classifiers in different types of features are listed.

- Acoustic Feature: Neural Net
- Artist Terms: Neural Net
- Lyrics: Logistic Regression
- Social: Tags Naïve Bayes

The results of these genre sub-classifiers generate confusion matrixes as shown in Figure 14. The best confusion matrix is expected to be a diagonal matrix.

The four sub-classifiers are combined based on the sub-classifier authority and the classification confidence. The resulting combined classifier is significantly better than each sub-classifier. Also, the combined classifier surpasses the SVM classifier using concatenated vectors from four data sources as shown in Figure 15. The classification accuracies of these classifiers are summarized in Table 3. The result is encouraging regarding to the result of genre classification task in



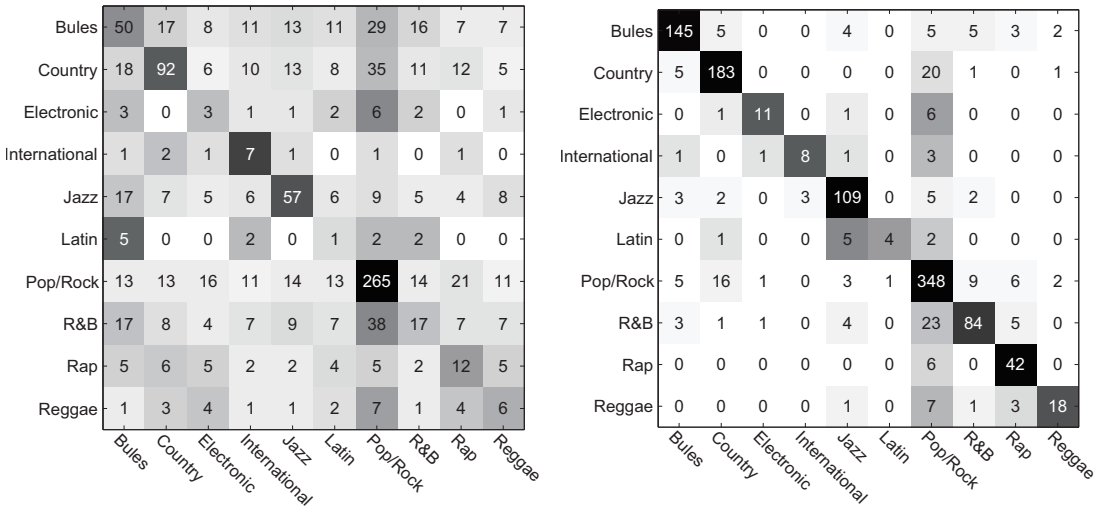
(a) Neural Net by audio features

(b) Neural Net by artist terms

(c) Logistic Regression by lyrics

(d) Naïve Bayes by social tags

Figure 14: Confusion matrixes of four sub-classifiers



(a) SVM by long vector (b) Combined classifier

Figure 15: Confusion matrixes by all data

MIREX [MIREX, 2009]. Furthermore, we apply the combined classifier to classify all of the songs in the MSD.

Table 3: 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 classifiers	**83.66**%

Chapter 6

Conclusion

This paper presented a novel approach in recommending songs one by one based on user behavior. The approach considered genre, recording year, freshness, favor and time pattern as factors to recommend songs. The evaluation results demonstrate that the approach is effective.

In further research, we can apply this technique to a music database in a server. Also other users' behavior can be applied to recommend songs for a user. We can mix recommendation of music in a local device and an online server data to overcome the issue of cold start and hence obtain new favorite songs.

Based on classifier authority and classification confidence, the combined classifier integrates sub-classifiers, which are good at classification of certain data sources. The combined classifier performs with higher accuracy than sub-classifiers and the SVM classifier using concatenated vectors.

REFERENCES

[Bertin-Mahieux et al., 2011] Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

[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*, 65:1509C1526.

[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*.

[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.

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

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

[Flexer et al., 2008] Flexer, A., Schnitzer, D., Gasser, M., and G., W. (2008). Playlist generation using start and end songs. In *9th International Conference on Music Information Retrieval*, pages 173–178.

[IMDB, 2012] IMDB (2012). [Online]. Available: http://www.imdb.com.

[Last.fm, 2012] Last.fm (2012). [Online]. Available: http://labrosa.ee.columbia.edu/millionsong/lastfm.

[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.

[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.

[Logan, 2004] Logan, B. (2004). Music recommendation from song sets. In *5th International Conference on Music Information Retrieval*, pages 425–428.

[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.

[MIREX, 2009] MIREX, A. G. C. (2009). [Online]. Available: http://www.music-ir.org/mirex/wiki/2009.

[musiXmatch, 2012] musiXmatch (2012). [Online]. Available: http://labrosa.ee.columbia.edu/millionsong/musixmatch.

[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.

[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*.

[RapidMiner, 2012] RapidMiner (2012). [Online]. Available: http://en.wikipedia.org/wiki/RapidMiner.

[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.

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

[Weka, 2012] Weka (2012). [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.