Evaluating Collaborative Filtering Algorithms for Music Recommendations on Chinese Music Data

By
Yifan He

An Undergraduate Senior Honors Thesis
Presented to the faculty
Of the University of North Carolina at Chapel Hill
In Partial fulfillment of the requirements
For the Honors Carolina Senior Thesis in the School of Information and Library Science

Chapel Hill
2021

Approved by:
Dr. Jaime Arguello, Thesis Advisor
Dr. Stephanie W. Haas, Reader
Dr. Ryan Shaw, Reader

Contents

CONTENTS **3**
LIST OF FIGURES **5**
LIST OF TABLES **6**
1. INTRODUCTION **7**
2. RELATED WORKS **12**
2.1 INTRODUCTION 12
2.2 THREE TYPES OF MUSICAL METADATA 12
2.3 MUSIC CONTENT-BASED METHODS 13
2.4 MUSIC CONTEXT-BASED METHODS 14
2.5 USER CONTEXT-BASED METHODS 16
2.5.1 Collaborative Filtering *16*
2.5.2 Explicit and Implicit User Feedback *16*
2.5.3 Memory-based Collaborative Filtering *17*
2.5.3.1 User-based Collaborative Filtering 18
2.5.3.2 Item-based Collaborative Filtering 19
2.5.4 Model-based Collaborative Filtering *21*
2.5.4.1 Matrix Factorization-based Collaborative Filtering 21
2.5.5 Users' Interactions with Recommender Systems *22*
2.6 CONCLUSION 23
3. METHODOLOGY **24**
3.1 DATASET 24
3.1.1 Parse Dataset *26*
3.1.2 Format Dataset for Recommender System *27*
3.2 MODELS 29
3.2.1 Notation Schema *31*
3.2.2 Baselines *32*
3.2.2.1 Min Predictor 32
3.2.2.2 Max Predictor 32
3.2.2.3 Normal Predictor 32
3.2.3 kNN-based Models *33*
3.2.3.1 *kNN* Basic 34
3.2.3.2 *kNN* With Normalization 34
3.2.3.2.1 *kNN* with *Means* 34
3.2.3.2.2 *kNN* with *Z-Score* 35
3.2.3.3 Similarity Metrics for *kNN* Models 36
3.2.3.3.1 Cosine Similarity 37
3.2.3.3.2 Mean Squared Difference 38
3.2.3.3.3 Pearson Correlation Coefficient 39
3.2.4 Slope One *40*
3.2.5 Matrix Factorization Models *42*

3.2.5.1 SVD 42
3.2.5.2 SVD++ 44
3.3 EVALUATIONS 46
3.3.1 Mean Squared Error (MSE) *46*
3.3.2 Root Mean Squared Error (RMSE) *46*
3.3.3 Mean Absolute Error (MAE) *47*
3.3.4 Fraction of Concordant Pairs (FCP) *47*
4. RESULTS AND DISCUSSION 49
4.1 BASELINES 52
4.2 *KNN*-BASED ALGORITHMS 55
4.3 SLOPE ONE ALGORITHM 57
4.4 MATRIX FACTORIZATION-BASED ALGORITHMS 58
4.5 LIMITATIONS 59
5. CONCLUSION AND FUTURE WORKS 61
BIBLIOGRAPHY 64
APPENDIX 69

List of Figures

Figure 1.1 Features of music content-based, music context-based and user context-based approaches.....8
Figure 2.1 User-based Collaborative Filtering Logic Flow.....19
Figure 2.2 Item-based Collaborative Filtering Logic Flow.....20
Figure 3.1 Interface of NetEase Cloud Music's Homepage.....25
Figure 3.2 Interface of a single playlist.....25
Figure 3.3 Sample MovieLens dataset format.....28
Figure 3.4 Logic flow for Slone One method.....41
Figure 3.5 Logic flow for SVD method.....43
Figure 4.1 Comparing all models on RMSE.....50
Figure 4.2 Comparing all models on MSE.....51
Figure 4.3 Comparing all models on MAE.....51
Figure 4.4 Comparing all models on FCP.....52
Figure 4.5 Comparing user-based kNN models on Cosine.....53
Figure 4.6 Comparing item-based kNN models on Cosine.....54
Figure 4.7 Comparing user-based kNN models on MSD.....54
Figure 4.8 Comparing item-based kNN models on MSD.....54
Figure 4.9 Comparing user-based kNN models on Pearson.....55
Figure 4.10 Comparing item-based kNN models on Pearson.....55

List of Tables

Table 3.1 Sample format of a Playlist 27
Table 3.2 Sample format of our dataset following MovieLens 29
Table 3.3 Models used in our experiment 30
Table 3.4 Notation Schema 31
Table 3.5 Similarity Metrics for kNN Models 36
Table 4.1 Experiment results for all models 49

1. Introduction

The amount of digital music available has overwhelmingly increased during the last few decades with the development of Internet and music technology. The use of digital music platforms has gradually replaced many traditional music consumption methods, such as tapes, CDs, and vinyl records. The explosive growth of digital music requires powerful knowledge management techniques and tools. Without these tools, users will face a vast music catalog that cannot be fully utilized. The research on Music Information Retrieval (MIR) is dedicated to solving these problems.

As a subfield of multimedia Information Retrieval, MIR is a highly dynamic and multidisciplinary field of research relates to various other research disciplines, involving researchers from the disciplines of musicology, library and information science, cognitive science, computer science, electrical engineering, and so on. Narrowing the focus on information retrieval (IR) and information representation related to music, we can distinguish three broad categories of strategies in terms of their data source: music content-based, music context-based and user context-based approaches. The music content-based approach extract music features from the audio signal itself, such as rhythm patterns, melodies, chords progressions, loudness, and so on. The music context-based approaches make use of text-based data related to music, such as the performer's background, the song's lyrics, images of album cover, or co-occurrence information derived from playlists. The user context-based approaches put focuses on users' personal data, such as their mood, physiological and spatial-temporal context, listening pattern and history, and so on. The following Figure 1.1 show their difference.

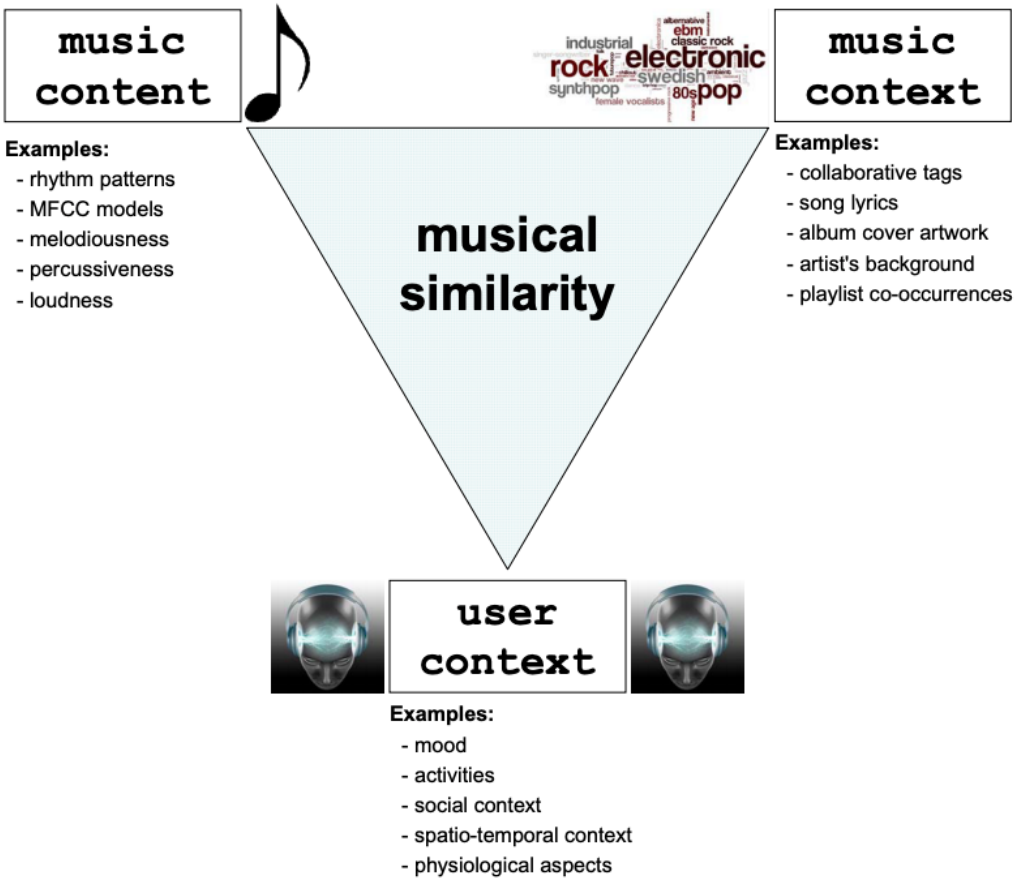

Figure 1.1 Features of music content-based, music context-based and user context-based approaches

In the user scenarios of digital music, many user requirements are proposed, including music identification, query by humming, auto music transcription and alignment, and so on. Among all these tasks, music recommendation is one of the most important and challenging tasks: recommending music for each user in a personalized way. MIR researchers use various music-related metadata to process and build algorithms to address this task. The state-of-the-art approaches to music recommendation are based on measuring music similarity between raw audio tracks, music-related metadata on the Web, and users' profiles, eliciting similar music. In these ways, all content-based audio features, text-based music-related metadata, and user-based community metadata can be used. By using different types of data, several approaches were developed respectively. Using user-based community metadata, the most used one is

Collaborative Filtering (CF), which analyzes one single user's behavior and compares it to other similar users' behaviors. The trick here is that it recommends new things based on the similarity between users' behaviors, not between items' intrinsic properties. The second method, using music content-based audio features, is the music content-based model. It analyzes to determine similar songs based on their similar acoustic patterns and audio features. The third method is the music text-based method. It uses music-related metadata originating from playlists, web pages, song lyrics, etc., extensively applied techniques from Information Retrieval and Natural Language Processing, such as TF-IDF weighting, Bag-of-Words representation, Part-of-Speech Tagging, and so on. The last method is called the hybrid approach, which combines two or more approaches of the previous three methods (Knees et al., 2015).

However, in today's industry, the music content-based and music context-based methods have not been employed very successfully in large-scale music repositories so far. Compare to the former two methods, it seems that user context-based methods especially collaborative filtering approaches have higher user acceptance and outperform the other two methods for music retrieval. It does not need to analyze the massive music-related data, but focuses on the users themselves, which can better realize personalized recommendations.

Despite user context-based collaborative filtering algorithms are widely studied and applied in the industry, their main applications are in e-commerce and video streaming media areas, performing tasks such as e-commerce product recommendations (e.g., Amazon), and movie recommendations (e.g., Netflix). Related datasets are also concentrated in these areas, such as the *Amazon product* dataset [1], *MovieLens* dataset [2], and *Netflix Prize* dataset [3]. Other music-related datasets such as *million song* dataset [4] and *Spotify* dataset [5] mainly focus on audio features and text metadata, which can be used in research directions of music content-based and music context-based methods. The only public user context-based music datasets that could be used to perform user rating-based collaborative filtering is the antique

R2-Yahoo! Music dataset [6], which was collected 15 years ago from 2002 to 2006, and the songs and artists are basically from English-speaking countries. Currently, there is no public user context-based dataset specially developed for Chinese music, and there is almost no past research using the Chinese music dataset to study the recommendation algorithms. MIR research has been developing in the U.S and other European countries for over 20 years, but it just gets started in China recent years.

Due to cultural differences, people from different countries and regions may have different consumption habits for music. For the same piece of music, people from English-speaking countries and Eastern countries may give different ratings. These cultural and customary differences may cause the collaborative filtering technology developed using English data sets to be unsuitable for the markets of Eastern countries. China is now the largest music consumer market, and people's demand for music recommendations is increasing. By studying collaborative filtering algorithms for Chinese music, it is possible to meet the needs of more people for music consumption, and at the same time create huge economic benefits.

Therefore, in this thesis, we plan to gather music-related information from Chinese music streaming platforms and build a new user rating-based dataset to evaluate the effectiveness of those algorithms. NetEase Cloud Music [9] is currently one of the largest music streaming providers in China, with over 300 million users and a music database containing over 10 million songs. We will crawl our data from their website, and more details are in section 3.1. Also, as the core of the recommendation system, the research of recommendation algorithms has received extensive attention from academia and industry for years. Score prediction is the core issue in the research of recommendation algorithms. Although there are endless research studies on various algorithms of recommendation systems, in the field of music recommendation, there is a lack of systematic comparison of various collaborative filtering

algorithms and evaluate their effectiveness. So, this thesis also plans to select a series of major CF algorithms and test them on our dataset.

MIR research on music recommendation benefits users since they could interact with digital music in a more convenient way. It also has great commercial value for music streaming providers including Spotify and Apple Music, especially for those companies operating in China since the data used in this thesis focus on Chinese music. By implementing MIR technologies to their products, music streaming companies can improve their users' experience. From the business point, better users' experience attracts more customers and improves users' loyalties to the product, which increases sales and profit.

In this thesis, I propose to explore user context-based methods for music recommendations. My research question is: Do our CF algorithms work on our Chinese Music dataset? And What are the differences between their performance? The research in this thesis has two contributions: first, developing a new dataset from music-related user data on Chinese music, as standing in 2021; second, comparing several collaborative filtering algorithms (Memory-based and Model-based) for music recommendation on our dataset. The remainder of this thesis organized as follows. In section 2, related works in the three directions of music content-based, music context-based and user context-based methods will be introduced, and the focus is on the user context-based. In section 3, the methods of developing our dataset and the models used in our experiment will be discussed. Section 4 describes the results and discussions of our experiment. Finally, Section 5 discuss our conclusion and the future work.

2. Related Works

2.1 Introduction

In this chapter, the introduction of the three general categories of artist-related data, music content-based, music context-based and user context-based data, mentioned in section 1 are presented. The purpose is to give an overview of the three research directions, to discuss related work, and to present the state-of-the-art in the corresponding areas. The structure of this chapter is as following: first, we first introduce the three types of metadata can be used; second, briefly go through selected research studies using music content-based and text-based methods. Third, go through some research studies using user context-based methods and collaborative filtering techniques used in this direction.

2.2 Three Types of Musical Metadata

Pachet (2005) distinguished three types of musical metadata:

(1) Acoustic metadata refers the digital signals obtained by analyzing the audio files.

(2) Cultural metadata is produced by users in culture environments.

(3) Editorial metadata refers to metadata manually annotated by the expert editors.

While acoustic metadata is a content-based method that focuses on extracting and using the attributes of the audio itself, cultural and editorial metadata focus on music-related textual data, including artists, genres, styles, labels, and users' reviews and ratings, and so on.

2.3 Music Content-based Methods

The concept of music similarities has traditionally been defined on the audio track level. It is calculated on the low-level and high-level audio-based features. Those features can be extracted by applying Digital Signal Processing (DSP) techniques. Those features are commonly known as content-based, audio-based, or signal-based. An overview of common content-based extraction techniques is presented in (Casey et al., 2008). There are vast number of existing literatures on the topic of audio feature extraction and similarity measurement between pieces of music.

In general, content-based features could be low-level representations that stem from the spectral centroid (Scheirer and Slaney, 1997), zero-crossing rate (Gouyon et al., 2000), bandwidth and band energy ratio (Li et al., 2001), amplitude envelope (Burred and Lerch, 2003). Alternatively, audio-based features may be aggregated from low-level properties, and then represent aspects on a high level of music understanding. Those high-level features usually aim at capturing rhythmic patterns and descriptors (Pampalk et al., 2002a, Dixon et al., 2003, Gouyon et al., 2004, Dixon et al., 2004), spectral properties in order to describe timbre (Foote, 1997, Logan, 2000, Logan and Salomon, 2001, Aucouturier and Pachet, 2004, Aucouturier et al., 2005, Mandel and Ellis, 2005), collaborative tagging (Aucouturier et al., 2007), and melodiousness (Pohle, 2009, Vikram and Shashi, 2017).

2.4 Music Context-based Methods

In this section, we review research studies that exploit textual representation of musical knowledge originating from playlists, web pages, and song lyrics, etc. Techniques from Information Retrieval (IR) and Natural Language Processing (NLP) were extensively applied, such as the *TF-IDF* weighting, *Bag-of-Words* representation, *Part-of-Speech* Tagging, and so on. One of the first approaches in this direction is in (Pachet et al., 2001), where they exploit radio station playlist and compilation CD databases and check the co-occurrences information between tracks and artists. A later research study by Baccigalupo et al. (2008) exploits playlists to derive artist similarity is on a web community.

Another source for artist similarity is the extremely large amount of available Web pages. Since the Internet reflects the opinions from lots of different people, interest groups, and companies, approaches to derive artist similarity from Web data incorporate the kind of "collective knowledge" and "wisdom of the crowd", and thus provide an important indication for the perception of music. Cohen and Fan (2000) enlarged the work of Shardanand and Maes (1995), and proposed the first work that crawled and filtered data from the Web to generate lists of similar music by different genres and artists. Whitman and Lawrence (2002) presented the first work dealing with free text metadata on the Web about musical artists for music recommendation engines. Much of their work is a combination of techniques in information retrieval applied to the music domain. They analyze and extract different term sets (unigrams, bigrams, noun phrases, artist names, and adjectives) from a set of about 400 artists by Part-of-Speech tagger from artist-related Web pages. Based on the term occurrences, term profiles are created for each artist. By this way, the similarity of artists is estimated by the term profiles between them. And then they generated a system to calculate the words co-occurrence of those artists by using TF.IDF and Gaussian Scoring Metric.

Collaborative tags provide another perspective towards similarity between music. A tag basically contains a short description about a specific aspect to the item. The more people label an item with the same tag, the more the tag is considered to be relevant to the item. Geleijnse et al. (2007) use tags from *Last.fm* [7], a music website where users could use tags to describe the music they listen to, and to generate a "tag ground truth" for artists similarities. They filter out redundant and useless tags and only use the set of tags associated with tracks by the artist of their choice. Then, the similarities between artists can be calculated by the number of overlapping tags. Compare with web-based text methods, tag-based methods have several advantages, including a more music-targeted and smaller vocabulary with less unrelated terms, and availability of descriptors for individual tracks rather than just artists.

The lyrics of songs can also be used to consider music similarities, since they usually represent information about the artist or the performer, such as cultural background, political orientation, and style of music. Logan et al. (2004) use song lyrics of tracks by 399 artists to determine their similarity. Mahedero et al. (2005) prove the usefulness of lyrics for four important tasks: language identification, structure extraction, thematic categorization, and similarity measurement. Other research studies do not explicitly aim at finding similarities in lyrical, but at revealing conceptual clusters (Kleedorfer et al. 2008), classifying songs into genres (Mayer et al., 2008), and mood categories (Hu et al., 2009).

2.5 User Context-based Methods

User feedback is another source to considering music similarity. Methods using this data source are also known as Collaborative Filtering (CF). To perform this similarity estimation, one must have access to a community and its activities. Therefore, CF methods are usually applied in real-world recommendation applications such as Amazon and Netflix.

2.5.1 Collaborative Filtering

Collaborative filtering is an effective technique in recommendation systems. It can be categorized into two main methods as memory-based (Neighborhood-based) and model-based (Latent factor models) collaborative filtering. From a general point of view, collaborative filtering refers to the process of exploiting large amounts of collaboratively generated data to filter items irrelevant for a given user from a repository. The aim is to retain and recommend those items that are likely a good fit to the taste of the target user.

2.5.2 Explicit and Implicit User Feedback

CF is characterized by large amounts of users and items and makes heavy use of users' taste or preference data, which expresses some kind of explicit rating or implicit feedback. Feedbacks on music items are sometimes given explicitly in the form of ratings. These ratings can be given on different levels or scales, including continuous natural values (e.g., between 1 and 100 score; 1-5/1-10 stars), binary ratings (e.g., thumbs up/down; like/unlike). Users' implicit feedback could be obtained from their actions during retrieval, such as browsing or by tapping. For example, users' browsing time, skipping a song, and purchase/consume history could be used as their implicit feedback.

Collected explicit ratings are usually represented in a user-item matrix R, where $r_{u,i} > 0$ indicates that user u has given a rating to item i (u has listen to i at least once), and higher values shows stronger preference. Given implicit feedback data, such a matrix can be constructed in a similar manner. For instance, a value $r_{u,i} > 0$ could also indicate that u has bought the CD, or has the song in the collection. $r_{u,i} < 0$ indicates that user u dislike item i (u has skipped song i while listening or u has rated i negatively), and $r_{u,i} = 0$ shows that there is no information available (or neutral opinion). When without additional evidence, a number of assumptions music usually be made in order to create such a matrix from implicit data (Ricci and Shapira, 2015).

The goal of such approaches is to "complete" the user-item matrix. Based on the completed matrix, we want to predict those unrated items' rating. Generally, there are two types of rating prediction approaches, memory-based and model-based collaborative filtering.

2.5.3 Memory-based Collaborative Filtering

Memory-based collaborative filtering, also called Neighborhood-based CF, operates directly on the full rating matrix, which is in the memory. Although this requirement usually makes them not very fast and resource-demanding, they are still widespread due to their simplicity, exploitability, and effectiveness (Desrosiers and Karypis, 2011). Some potential disadvantages of memory-based collaborative filtering include scalability and sensitivity to data sparseness issue (Lemire and Maclachlan, 2005).

Shardanand and Maes (1995) proposed the personalized music recommendation system *RINGO*, by using social information filtering. The system maintains user profiles on their interests (positive and negative attitudes) towards specific music. And then it compares users'

profiles to infer their degree of similarity. Finally, it can recommend users music with the music in their similar users' profiles.

Celma (2009) provides a detailed discussion of collaborative filtering methods for music recommendation based on real-world examples from the music field. In Celma's work, Memory-based CF systems handles two types of similarity relationships: item-to-item similarity (songs or artist), and user-to-user similarity. The similarity between items can be calculated by comparing each *N*-dimensional row vectors, where *N* is the total number of users. The similarity between users can be calculated by comparing the corresponding *M*-dimensional column vectors, where *M* is the total number of items.

For vector comparison, cosine similarity and Pearson's correlation coefficient are popular choices. For example, Slaney and White (2007) analyze 1.5 million user ratings given by 380,000 users from the *Yahoo! music* service and obtain the similarity of music piece by comparing normalized rating vectors on the items and calculating their respective cosine similarities.

2.5.3.1 User-based Collaborative Filtering

The user-based CF approach recommends items to the target user from his/her similar users. For example, as seen in the Figure 2.1, in this case, user 1 and user 3 have similar music preferences, so they are considered as a neighbor to each other. Since user 1 gives a like to item A, the user-based CF system would recommend A to user 3. User-based CF uses users' rating scores on items to consider their similarities and finds their *k* nearest neighbors. Then, the system can make predictions based on weighted averaging score by combining all neighbors' ratings.

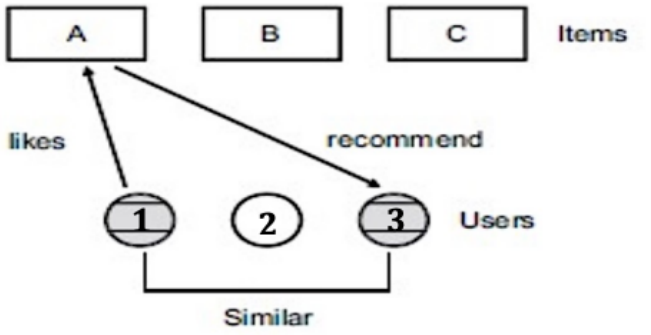

Figure 2.1 User-based Collaborative Filtering Logic Flow

To make our ideas into equation, a predicted rating $r'_{u,i}$ for user *u* and an item *i* is calculated by finding K_u, the set of *k* nearest neighbors according to their rating preference with *u*, and combine their ratings for item *i* (Konstan et al., 1997):

$$r'_{u,i} = \frac{\sum_{g \in K_u} sim(u,g)(r_{g,i} - \bar{r}_g)}{\sum_{g \in K_u} sim(u,g)} + \bar{r}_u$$

Where $r_{g,i}$ denotes the rating of user *g* given to item *i*, $\bar{r}_u$ is the average rating of user *u*, and *sim(u,j)* is a weighting factor that corresponds to the similarity to the neighboring users.

2.5.3.2 Item-based Collaborative Filtering

The item-based CF approach recommends items by checking similarities between the items that are already associated with the user. For example, as seen in the Figure 2.2, in this case, item A and item C are similar, so they are considered as a neighbor to each other. Since the user gives a like to item A, the item-based CF system would recommend C to the user. Item-

based CF analyzes a set of items that the target user has already evaluated to calculate the similarities between all items and the target item. Then, the system can make predictions for target items by comparing the user's previous preferences on the set.

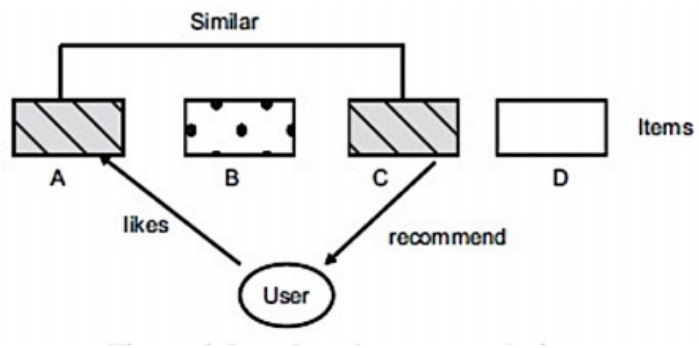

Figure 2.2 Item-based Collaborative Filtering Logic Flow

While user-based CF identifies neighboring users by examining the rows of the user-item matrix, item-based CF operates on the columns to find similar items. Predictions $r'_{u,i}$ are then made by finding the set of *k* most similar item to *i* that have been rated by *u* and combining the ratings. Since the number of rated items from each user is usually smaller than the total number of items, item-to-item similarities can be pre-calculated and cached. In many real-world recommender systems, item-based CF is thus chosen over user-based CF for performance reasons (Linden et al., 2003). For determining nearest neighbors (*kNNs*), in both user-based and item-based CF, cosine similarity or "cosine-like measure" are utilized (Nanopoulos et al., 2009).

2.5.4 Model-based Collaborative Filtering

There are many model-based approaches to collaborative filtering, in which some are based on Linear Algebra, such as Matrix Factorization, Principal component analysis (PCA), and Eigenvectors; some others use techniques derived from Artificial Intelligence, such as Bayes methods, Latent Classes, and Neural Networks; some others are based on clustering (Drineas et al., 2002). Compared with memory-based methods, model-based algorithms are usually preforming faster at query time, although they might need expensive learning and updating phases. In our thesis, we focus on the Matrix Factorization-based methods.

2.5.4.1 Matrix Factorization-based Collaborative Filtering

Model-based CF methods that build upon latent factors representation are obtained by learning factorization models of the rating matrix, e.g., (Hofmann, 2004), (Koren et al., 2009). Compare with memory-based CF methods, matrix factorization-based CF have a few advantages, including higher accuracy, shorter processing time since they avoid the massive calculations directly on the rating matrix like *kNN* methods, and creates more explicit and compact representations. The results of the *KDD-Cup* 2011 competition shows that matrix factorization methods can significantly improve prediction accuracy in the music field (Dror et al., 2011).

The assumption under the matrix factorization-based CF is that the observed ratings are the result of a number of latent factors of user characteristics (e.g., preference or taste, "personality") and item characteristics (e.g., genre, mood, instrumentation). The purpose of matrix factorization is trying to estimate those hidden parameters from users' ratings. These parameters should explain the observed ratings by characterizing both users and items in an ℓ-dimensional space of factors. These computations derived factors basically describe the variance in the rating data and may not be human interpretable (Ricci and Shapira, 2015).

In the matrix factorization model, the rating matrix R can be decomposed into two matrices W and H such that $R = WH$, where W relates to users and H to items. In the latent space, both users and items are represented as ℓ-dimensional vectors. Each user u is represented as a vector $w_u = R^{\ell}$, and each item i as a vector $h_i = R^{\ell}$. $W = [w_1 \dots w_f]^T$ and $H = [h_1 \dots h_f]^T$ are thus $f \times \ell$ and $\ell \times n$ matrices respectively. Entries in w_u and h_i measure the relatedness to the corresponding latent factor. These entries can be negative. The number of latent factors is chosen such that $\ell \ll n, m$, which yields a significant decrease in memory consumption in comparison to memory-based CF. Also rating prediction is much more efficient once the model has been trained by simply calculating the inner product $r'_{u,i} = w_u{}^T h_i$. More generally, after learning the factor model from the given ratings, the complete matrix $R' = WH$ contains all ratings predictions for all user-item pairs.

2.5.5 Users' Interactions with Recommender Systems

Other research studies explore users' interaction with their recommender systems. Pohle et al. (2007) present a user interface that allows users to choose musical concepts tags they extracted from *last.fm* [7], and recommend them music match with the concepts tags they choose. Knees and Widmer (2008) present an approach to adapt user preferences on music by incorporating relevance feedback. Mesnage et al. (2011) build a social music recommender system by investigating *Facebook*, and recommend music based on users' friend relationships and their interactions with other users.

2.6 Conclusion

We have gone through the state-of-the-art methods of music content-based, music context-based, and user context-based in music recommendation tasks. In contrast to the content-based and text-based approaches reviewed in sections 3 and 4, collaborative filtering techniques used in user context-based methods could outperform the previous two methods in today's industry since they do not require any additional metadata and calculations that related to the music pieces themselves. Due to the nature of rating matrix, similarities can be directly calculated without correlating the occurrence of metadata with actual items. However, these CF approaches are sensitive to factors such as popularity biases and data sparsity.

In the following section, we discuss the process of our dataset preparations, and introduce the CF models will be used in our experiment.

3. Methodology

3.1 Dataset

The raw dataset was crawled from NetEase Cloud Music [9], one of the largest music streaming platforms in China, with over 300 million users and a music database containing over 10 million songs. On this platform, users can search and listen to a large number of songs. When a user encounters a song he likes, he/she can click the “Like” button and add it to his/her playlist. Users can add tags to their playlists to illustrate the style of the songs in the playlist, such as "Mandarin", "Popular", "Electronic", etc. Each user can create multiple playlists, and at the same time set the playlists to be public so that they can be accessed by other users. Accessing the playlists of other users is the major way for users to explore music. On the homepage of the platform, popular playlists are displayed for other users to access. Users can also subscribe other users' playlists to get the latest music trends. The homepage interface of NetEase Cloud Music is shown on Figure 3.1. The interface of a single playlist is shown on Figure 3.2.

Interestingly, NetEase Cloud Music has already provided users with a "Daily Song Recommendation" playlist function. According to the user's behavior, this playlist will automatically generate 20 songs that the user may like every day. The company may use more indicators to help them measure users' behavior, such as users remove an item from the playlist, user skipping a song, and the time they spent on a particular page and the number of clicks. However, we cannot access this part of the information, nor can it be obtained through web crawling. The only information we can obtain is the playlist of each user and the songs contained in it. Since we want to focus on Chinese Music, we only crawl the playlists that contain Mandarin tags.

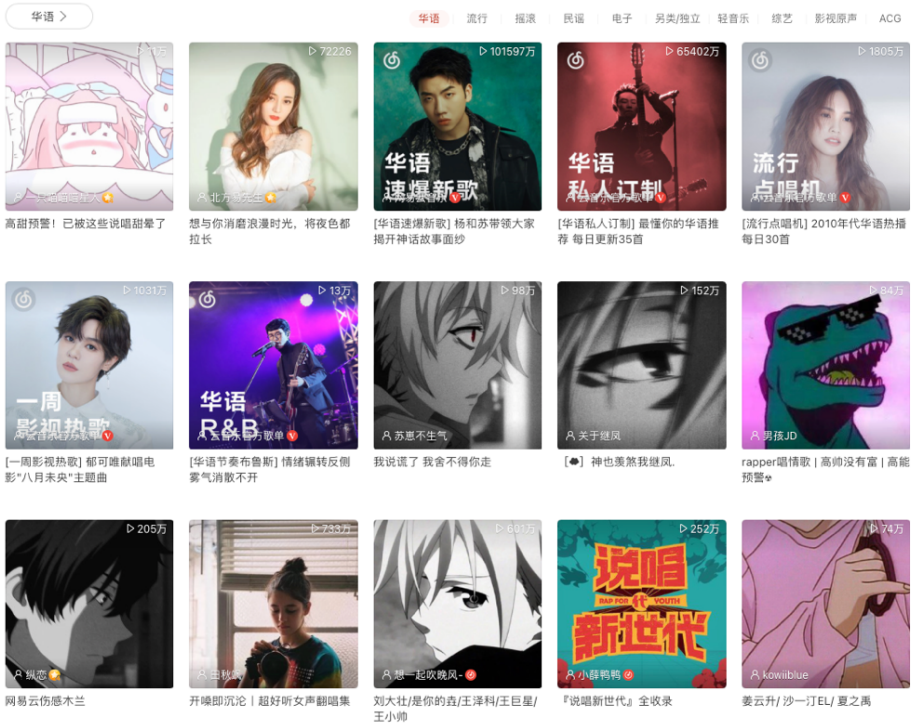

Figure 3.1 Interface of NetEase Cloud Music's Homepage

Figure 3.2 Interface of a single playlist

Our dataset contains over 50,000 public playlists contain Mandarin tag of corresponding users and more than 200,000 songs. The format is in JSON, and the size of our file *playlist.detail.all.json* is 16.08 Gigabyte. In addition to the metadata of the playlist, each playlist also includes all the tracks in the playlist and their metadata. The sample format of each playlist and each song are listed in the Appendix A and Appendix B.

3.1.1 Parse Dataset

Since the raw dataset contains too many playlists and songs, it is difficult to process this amount of information on a personal computer. Also, some of the playlists in the dataset only have few songs, which cause the problem of data sparsity. So, we need to parse the raw dataset.

We find that the playlists with more subscribers, in general, have more songs. Those users are considered active users of the platform. So, we only take those playlists with more than 100 subscribers (*subscribedCount > 100*). The total number of playlists that fit the requirement is 1076, containing 49774 songs. Also, the raw dataset contains too much information, including *createTime*, *updateTime*, *description*, *URLs* and many tags, etc. This part of the information is not very helpful to our research, so we decided to extract only a few of the most important elements and convert the raw dataset into a simplified dataset. For each playlist, only four dimensions of the information including playlist name, category, playlist id, and the number of subscribers is extracted. For each song, only four dimensions of the information including song id, song title, singer, and song popularity is extracted. For each playlist and the songs contained in it, we organize them into the following Table 3.1:

Table 3.1 Sample format of a Playlist

Playlist	##流行,华语##57364826##3572
Track 1	28643203::花房姑娘::崔健::57.0
Track 2	407007442::末班车::信::46.0
Track 3	33162226::友情岁月::李克勤::85.0
Track 4	28387594::无赖::林忆莲::74.0
Track …	.....

3.1.2 Format Dataset for Recommender System

Users' rating to an item is an explicit feedback. For some datasets, they contain such information, so those rating could be used in the matrix. Nonetheless, for other datasets, they do not contain rating information, just as the dataset in our experiment. In this case, we need to use users' implicit feedback as their rating information. For example, users' browsing, or purchase history could be used as their rating.

Since the playlist is a collection behavior, which shows the user's interest in the songs in the playlist, the songs in each user's playlist can be regarded as a positive feedback of the user for the songs in the playlist. However, users' negative feedback on songs cannot be directly reflected in the user's collection behavior. This is determined by user's behavior when consuming music. When they dislike a song, the usual behavior is to skip it. Although NetEase Cloud Music provides users with the function of disliking songs, which is a clear negative feedback, users usually choose to skip the current song instead of clicking the dislike button for the song they dislike, resulting in a small amount of data in this part. Also, the data of users' dislike songs are not displayed on the playlist page and cannot be obtained through crawlers.

Based on these considerations, we assume that the songs that are not collected by users reflect their dislike of these songs to a certain extent. Although it is a bit arbitrary, we decided to directly sample the songs outside of each user's playlist as a negative feedback for them. This creates a binary matrix, where "1" stands for "rated and like", and "0" for "dislike". Although songs may not be selected into the playlist is because users have not listened to these songs, we believe this is the best solution to approach our dataset based on our knowledge. The entire experiment was performed under this assumption, which could be one of our limitation, and further discussion is in the section 4.

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
5	1	70	3.0	964982400
6	1	101	5.0	964980868
7	1	110	4.0	964982176
8	1	151	5.0	964984041
9	1	157	5.0	964984100

Figure 3.3 Sample *MovieLens* dataset format

For mainstream python recommendation system frameworks, the most basic data format supported is the *MovieLens* dataset [2] ,and its data format is user-item-rating-timestamp. Its format is shown in Figure 3.3. In order to use the framework, we decide to process our data into the same format. For rating information, we set the positive feedback to score 1.0, and the negative feedback to 0.0. For timestamp information, since our data do not have this part of information, and it actually have no effect on our algorithms, we simply give them the same value 1300000. Since we find the average items contained in all the playlists is 150 items, we

construct 150 negative sample for each user and generate the files *music_format_neg.txt*. Then we merge the data of *music_format_neg.txt* into *music_format.txt* to get the final model training data *music_format_full.txt*. In the modeling process, we randomly split the train set at the ratio of 75%:25%, and take 25% as the test set. The final data format is as the following Table 3.2:

Table 3.2 Sample format of our dataset following *MovieLens*

user_id, song_id, score, time_stamp
392991828,33891487,1.0,1300000
392991828,31168297,1.0,1300000
392991828,101085,1.0,1300000
392991828,407761300,1.0,1300000
392991828,29738501,0.0,1300000
392991828,48365894,0.0,1300000
.....

3.2 Models

This paper uses the python library *Surprise* (Hug, 2020) in the modeling process of the recommender system. *Surprise* is a Python library for building and analyzing rating prediction algorithms. It was designed to closely follow the scikit-learn API.

We use the models listed in Table 3.3 to predict the user-item ratings and we will discuss their performance in the section 4.

Table 3.3 Models used in our experiment

Category	Name	Description
Baseline	Min Predictor	A baseline algorithm predicting a minimum rating based on the train set.
Baseline	Max Predictor	A baseline algorithm predicting a maximum rating based on the train set.
Baseline	Normal Predictor	A baseline algorithm predicting a random rating based on the distribution of the train set.
Memory-based (Neighborhood-based)	*kNN* Basic	A basic *kNN* CF algorithm.
Memory-based (Neighborhood-based)	*kNN* with Means	An improved *kNN* algorithm, taking the mean normalization of each user into account.
Memory-based (Neighborhood-based)	*kNN* with Z-Score	An improved *kNN* algorithm, taking the z-score normalization of each user into account.
Model-based	*Slope One*	A simple yet accurate CF algorithm.
Model-based (Matrix Factorization)	*SVD*	An algorithm to identify latent semantic factors by decomposing matrix.
Model-based (Matrix Factorization)	*SVD++*	An improved *SVD* algorithm, taking implicit ratings into account.

3.2.1 Notation Schema

Our notation schema is listed in following Table 3.4.

Table 3.4 Notation Schema

R	the set of all ratings.
$R_{train}, R_{train}, \hat{R}$	the training set, the test set, and the set of predicted ratings.
U	the set of all users. *u* and *v* denote users.
I	the set of all items. *i* and *j* denote items.
U_i	the set of all users that have rated item *i*.
U_{ij}	the set of all users that have rated both items *i* and *j*.
I_u	the set of all items rated by user *u*.
I_{uv}	the set of all items rated by both users *u* and *v*.
r_{ui}	the true rating of user *u* for item *i*.
$\hat{r}_{ui}$	the estimated rating of user u for item *i*.
b_{ui}	the baseline rating of user u for item *i*.
u	the mean of all ratings.
u_u	the mean of all ratings given by user *u*.
u_i	the mean of all ratings given to item *i*.
σ_u	the standard deviation of all ratings given by user *u*.
σ_i	the standard deviation of all ratings given to item *i*.
$N_i^k(u)$	the k nearest neighbors of user *u* that have rated item *i*.
$N_u^k(i)$	the k nearest neighbors of item *i* that are rated by user *u*.

3.2.2 Baselines

We propose three baseline models to evaluate the effectiveness of all other models. The idea here is our algorithms must at least perform better than our baselines to prove they are doing a predictive job. The Min Predictor would always output 0, which means dislike; and the Max Predictor would always output 1, which means like. The Normal Predictor predicts a random number based on the normal distribution of our training set, which would be a fraction number between 0 and 1.

3.2.2.1 Min Predictor

Min Predictor predicts a minimum rating based on the train set, which is 0 in our dataset.

$$\hat{r}_{ui} = \min r_{ui}$$

3.2.2.2 Max Predictor

Max Predictor predicts a maximum rating based on the train set, which is 1 in our dataset.

$$\hat{r}_{ui} = \max r_{ui}$$

3.2.2.3 Normal Predictor

The Normal Predictor Baseline algorithm predicts a random rating based on the distribution of the train set. The prediction$\hat{r}_{ui}$ is generated from a normal distribution $N(\hat{u}, \hat{\sigma}^2)$ where $\hat{u}$ and $\hat{\sigma}$ are estimated from the training data using Maximum Likelihood Estimation.

$$\hat{\mu} = \frac{1}{|R_{train}|} \sum_{r_{ui} \in R_{train}} r_{ui}$$

$$\hat{\sigma} = \sqrt{\sum_{r_{ui} \in R_{train}} \frac{(r_{ui} - \hat{\mu})^2}{| R_{train} |}}$$

3.2.3 *kNN*-based Models

The k-Nearest-Neighbors (*kNN*) is a non-parametric classification and case-based learning method, which keeps all the training data for classification (Ricci and Shapira, 2015). It is simple but effective in many cases. For a user u to be classified, its k nearest neighbors are retrieved, and this forms neighborhoods of u. The *kNN*-based algorithm automates the general principles of word-of-mouth, in which people consider and rely on the opinions of other like-minded people to evaluate items. It mainly including user-based and item-based two directions.

In the user-based CF scenario, when user A needs personalized recommendations, the system would first find other users who have similar interests with A, and then recommend those items that those user likes but that the user A has never known. The user-based CF algorithm mainly includes two steps: First, investigate the set of users with similar interests to the target users, and find its k nearest neighbors; Second, find items in those users' collections and not in the collection of the target user, and recommend them to the target users.

The item-based CF is the most widely used algorithm in the industry, recommending similar items to the users they previously liked. The item-based collaborative filtering algorithm mainly includes two steps: First, calculate the similarity between all items in the set with the target item, and find its k nearest neighbors; Second, generate recommendations for the user based on the similarity of items.

In all the *kNN* algorithms performed in our experiment, we set their k value equals 40.

3.2.3.1 *kNN* Basic

A basic form of *kNN* algorithm. The prediction $\hat{r}_{ui}$ is set as:

$$\text{User-based: } \hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

$$\text{Item-based: } \hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} \text{sim}(i,j)}$$

3.2.3.2 *kNN* With Normalization

Normalization is the process of adjusting values measured on a different scale to a common scale. It compensates for the difference in users' behavior by adjusting the rating scale, and make the range comparable or same with other users' ratings. Without normalization, our data would be unscaled and hence highly intricate to calculate and compare with other parameters (Pandey et al., 2017). There are many normalization techniques, including feature scaling, coefficient of variation, studentized residual, standard score (Z-Score), etc. In this thesis, we use Mean-centering and Z-Score to Normalize (Breese et al., 1998).

3.2.3.2.1 *kNN* with *Means*

kNN with Means is an improved *kNN* algorithm, in which the mean ratings of each user or item are considered. The idea of centering the mean is to determine whether a rating is positive or negative by comparing it with the mean rating. In user-based CF, the raw rating r_{ui} is

transformed into $h(r_{ui})$ by subtracting the average of the rating $\overline{r_u}$ to r_{ui}. In item-based CF, the $h(r_{ui})$ is done in the similar manner.

$$h(r_{ui}) = r_{ui} - \overline{r_u}$$

In our experiment, the prediction $\hat{r}_{ui}$ is set as:

User-based:
$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Item-based:
$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) \cdot (r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} \text{sim}(i,j)}$$

3.2.3.2.2 *kNN* with *Z-Score*

kNN with Z-Score is an improved *kNN* algorithm, considering the Z-Score normalization of each user or item. The Z-Score normalized each score to its number of standard deviations that it is distant from the mean score. While mean-centering eliminates the drift caused by the different perceptions of the average rating, Z-Score also takes into account the difference in each individual scales. In user-based CF, $h(r_{ui})$ equals to mean-centered r_{ui} divided by the standard deviation σ_u. In item-based, the $h(r_{ui})$ is done in the similar manner.

$$h(r_{ui}) = \frac{r_{ui} - \overline{r_u}}{\sigma_u}$$

In our experiment, the prediction $\hat{r}_{ui}$ is set as:

User-based:
$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v) / \sigma_v}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Item-based:
$$\hat{r}_{ui} = \mu_i + \sigma_i \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) \cdot (r_{uj} - \mu_j) / \sigma_j}{\sum_{j \in N_u^k(i)} \text{sim}(i,j)}$$

3.2.3.3 Similarity Metrics for *kNN* Models

The similarity weights play an important role in neighborhood-based CF methods. They provide different methods to give these neighbors more or less weight in the prediction.

For *kNN* Basic, *kNN* with Means, and *kNN* with Z-Score, we use the following three metrics in Table 3.5 to calculate their similarity for both user-based and item-based methods.

Table 3.5 Similarity Metrics for *kNN* Models

Cosine Similarity	Calculate the Cosine Similarity for both user-based and item-based *kNN* models.
Mean Squared Difference (MSD)	Calculate the Mean Squared Difference for both user-based and item-based *kNN* models.
Pearson Correlation Coefficient	Calculate the Pearson Correlation Coefficient for both user-based and item-based *kNN* models.

3.2.3.3.1 Cosine Similarity

Cosine similarity is one of the most popular metrics used in Information Retrieval. It treats objects as vectors in an *N*-dimensional vector space. In the following equation, cosine similarity investigates the angle between two vectors, item *i* and item *j*. $R_{u,i}$ is the rating of item *i* given by user *u*. $R_{u,j}$ is the rating of item *j* by user *u*. *n* is the total number of all ratings to item *i* and item *j* (Billsus and Pazzani, 1998).

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|^2\|\vec{j}\|^2} = \frac{\sum_{u=1}^{n} R_{u,i}R_{u,j}}{\sqrt{\sum_{u=1}^{n} {R_{u,i}}^2}\sqrt{\sum_{u=1}^{n} {R_{u,j}}^2}}$$

Two vectors are considered similar if the angle between them is small. When the angle reaches 0 digress, *sim(i,j)* = 1, indicating we can regard them as identical. When the angle reaches 180 digress, *sim(i,j)* = -1, indicating we can regard them as opposite. If the angle is 90 degrees, *sim(i,j)* = 0, meaning the two vectors are irrelevant. In our cases, *sim(i,j)* ranges from 0 to 1.

In our experiment, we calculate the cosine similarity between all pairs of users and items. The cosine similarity for users is defined as:

$$\text{cosine\_sim}(u,v) = \frac{\sum_{i\in I_{uv}} r_{ui}\cdot r_{vi}}{\sqrt{\sum_{i\in I_{uv}} r_{ui}^2}\cdot\sqrt{\sum_{i\in I_{uv}} r_{vi}^2}}$$

The cosine similarity for items is defined as:

$$\text{cosine\_sim}(i, j) = \frac{\sum_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{uj}^2}}$$

3.2.3.3.2 Mean Squared Difference

Mean Squared Difference (MSD) evaluates the similarity between two users *u* and *v* as the inverse of the average squared difference between the rating given by *u* and *v* on the same items (Shardanand and Maes, 1995).

$$\text{MSD}(u, v) = \frac{|\mathcal{I}_{uv}|}{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - r_{vi})^2}$$

We calculate the Mean Squared Difference similarity between all pairs of users or items. The MSD for users is defined as:

$$\text{msd\_sim}(u, v) = \frac{1}{\frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2 + 1}$$

The MSD for items is defined as:

$$\text{msd\_sim}(i, j) = \frac{1}{\frac{1}{|U_{ij}|} \cdot \sum_{u \in U_{ij}} (r_{ui} - r_{uj})^2 + 1}$$

3.2.3.3.3 Pearson Correlation Coefficient

Pearson correlation coefficient is one popular method used in collaborative filtering tasks. It measures the tendency of two number series, paired up one-to-one and move together (Deshpande and Karypis, 2004). The Pearson correlation coefficient can be considered as an improved cosine similarity with mean-centered (Ricci and Shapira, 2015). As in the following equation, it measures the linear correlation between two vectors item *i* and item *j*.

$$sim(i,j) = \frac{\langle R_{u,i} - A_i, R_{u,j} - A_j \rangle}{\left\| R_{u,i} - A_i \right\| \left\| R_{u,j} - A_j \right\|} = \frac{\sum_{u=1}^{n} (R_{u,i} - A_i)(R_{u,j} - A_j)}{\sqrt{\sum_{u=1}^{n} (R_{u,i} - A_i)^2} \sqrt{\sum_{u=1}^{n} (R_{u,j} - A_j)^2}}$$

n is the total number of all ratings given to *i* and *j*. $R_{u,i}$ is the rating of item *i* given by user *u*, $R_{u,j}$ is the rating of item j given by user *u*. A_i is the average rating of item *i* for all the co-rated users, and A_j is the average rating of item *j* for all the co-rated users. When two the vectors are in a high level of similar tendency, *sim(i,j)* is close to 1; when two vectors are in a low level of similar tendency, *sim(i,j)* is close to 0. When two vectors have an opposite tendency, *sim(i,j)* is -1. In our cases, *sim(i,j)* ranges from 0 to 1.

In our experiment, we calculate the Pearson correlation between all pairs of users and items. The Pearson correlation for users is defined as:

$$\text{pearson\_sim}(u,v)=\frac{\sum_{i\in I_{uv}}(r_{ui}-\mu_u)\cdot(r_{vi}-\mu_v)}{\sqrt{\sum_{i\in I_{uv}}(r_{ui}-\mu_u)^2}\cdot\sqrt{\sum_{i\in I_{uv}}(r_{vi}-\mu_v)^2}}$$

The Pearson correlation for items is defined as:

$$\text{pearson\_sim}(i,j)=\frac{\sum_{u\in U_{ij}}(r_{ui}-\mu_i)\cdot(r_{uj}-\mu_j)}{\sqrt{\sum_{u\in U_{ij}}(r_{ui}-\mu_i)^2}\cdot\sqrt{\sum_{u\in U_{ij}}(r_{uj}-\mu_j)^2}}$$

3.2.4 Slope One

Slope One algorithm is a linear and personalized algorithm based on the rating difference between different items to predict users' rating of items. Compared with other similar algorithms, it has several advantages: easy to implement, updateable on the fly, efficiency at query time, expect little information from users, and the relatively high accuracy of recommendation (Lemire and Maclachlan, 2005).

The slope one method considers both information from all users rated the same item and from all items rated by the same user. It follows the intuitive principle of the “popularity differential” between items and users. The way it measures such popularity differential is by simply subtract the average rating of the two items. In turn, this difference could be used to

predict the rating by another user on the item. For example, given two user A and B, two item I and J in the following Figure 3.4.

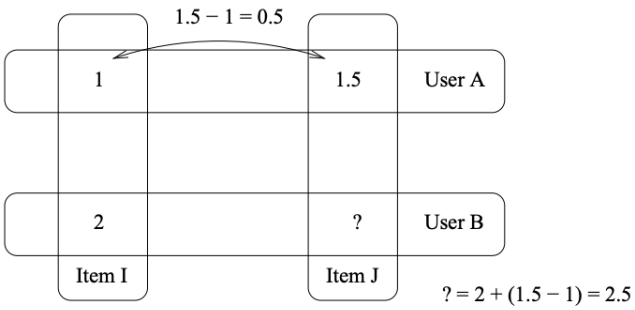

Figure 3.4 Logic flow for *Slone One* method

User A gives item I a rating of 1, item J a rating of 1.5, while user B gave I a rating of 2. The rating difference between I and J gave by User A is 0.5, so we simply use this value difference to calculate that user B will give item J a rating of 2 + 0.5 = 2.5.

Formulating this idea into equation, we first calculate the average deviation of item *i* relative to item *j* as:

$$\text{dev}(i,j) = \frac{1}{|U_{ij}|}\sum_{u\in U_{ij}} r_{ui} - r_{uj}$$

Note that those users not containing both u_i and u_j would not be included in the summation. Then according to the rating deviation between items and the user's historical rating, we predict the user's rating of unrated items. The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu_u + \frac{1}{|R_i(u)|}\sum_{j\in R_i(u)} \text{dev}(i,j)$$

$R_i(u)$ is the set of relevant items, for example, the set of items *j* rated by *u* that also have at least one common user with *i*.

3.2.5 Matrix Factorization Models

Matrix Factorization methods consider both items and users as vectors of factors inferred from the rating patterns. Recommendations are made based on high correspondence between item and user. Matrix Factorization methods have become popular in recent years since they have good scalability and predictive accuracy. Also, they provide more flexibility for modeling various kinds of real-life situations. Matrix factorization models map users and items to an *f*-dimensional joint latent factor spaces, in order to model user-item interactions as inner products in these spaces. The latent factor space attempts to explain ratings by characterizing both items and users through factors that are automatically inferred from users' feedback (Koren et al., 2009).

3.2.5.1 SVD

Singular Value Decomposition (*SVD*) methods can identify latent semantic factors. The key idea of *SVD* is to find lower-dimensional features space, where the new features can represent the "concepts" and the "strength" of each concept in the context of the collections (Wall et al., 2003). *SVD* algorithms work in the following process (Figure 3.5): first decompose a given matrix *A* into $A = U\lambda V^T$. By decomposing the $n \times m$ matrix A, we can obtain $n \times r$ matrix U ((represent the concepts and items), $r \times r$ matrix λ (represent the strength of each concept), and $m \times r$ matrix V (represent the features and concepts).

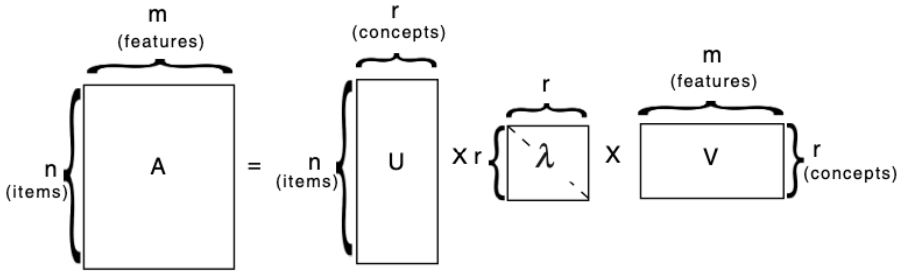

Figure 3.5 Logic flow for *SVD* method

Converting the idea into algorithms, each item i is associated with a vector $q_i \in R^f$, and each user u is associated with a vector $p_u \in R^f$. For item i, the elements of q_i measure the extent to which the item possesses those factors, positive or negative. For user u, the element p_u measures the extent of interest the user has in items that are high on the corresponding factors, positive or negative. The resulting dot product, ${q_i}^T p_u$, captures the interaction between user u and item i, that is, the user's overall interest in the item's characteristics (Ricci and Shapira, 2015). Thus, a rating is predicted as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

In order to estimate all the number unknown, the regularized squared error is minimized as the following equation:

$$\sum_{r_{ui} \in R_{train}} \left(r_{ui} - \hat{r}_{ui}\right)^2 + \lambda\left(b_i^2 + b_u^2 + \| q_i \|^2 + \| p_u \|^2\right)$$

The constant λ that controls the degree of regularization is usually determined by cross-validation. Minimization is usually performed by stochastic gradient descent or alternating least squares. In *Surprise*, the minimization is performed through the stochastic gradient descent as followings:

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$$

$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u)$$

$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i)$$

Sarwar et al. (2002) mentioned one of the advantages of *SVD* is that when calculating approximated decomposition, it is incremental so that new users and ratings can be accepted without having to recalculate the model that built from previous data.

3.2.5.2 SVD++

Koren (2008) proposes *SVD++*, which improved *SVD*' prediction accuracy by also considering users' implicit feedback, thereby providing another indication of user preferences. This implicit feedback can stem from the user's browsing history or from listening events and is expressed in binary form.

SVD++ is especially helpful for cases when users provide more implicit feedback than explicit feedback. Even in cases where implicit feedback is absent, it can capture a significant amount of information by considering which items users have rated, regardless of their rating value (Ricci and Shapira, 2015). This led to a second latent factor, which optimized in conjunction with those latent factors that modeling the explicit ratings. Each item *i* to a factor vector $y_i \in R^f$, those new latent factors are used to characterize users based on the set of items they have rated. The equation is shown as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

I_u is the set contains all items rated by user u, and y_j is the new set of item factors that capture implicit ratings. The implicit rating describes the fact that the user u rated the item j, and has nothing to do with the actual rating value. The bias b_u and factors p_u are assumed to be zero if user u is unknown. The same rules applied for item i with b_i, q_i, and y_i.

In our experiment, for both *SVD* and *SVD++*, the learning rates γ are set to 0.005, and the regularization terms λ are set to 0.02.

3.3 Evaluations

We use four evaluation metrics for comparing the performance of all algorithms discussed in section 3.2 on our dataset: MSE, RMSE, MAE, and FCP.

3.3.1 Mean Squared Error (MSE)

Mean Squared Error (MSE) is used to compare the predicted value with the real preference value a user has assigned to an item.

$$\text{MSE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2$$

Where $\hat{R}$ is the total number of ratings over all users, r_{ui} is the predicted rating for user u on item i, and $\hat{r}_{ui}$ is the actual rating.

3.3.2 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is one of the most popular metrics used in evaluating the accuracy of predicted ratings. It is becoming more popular since it is the metric used in the *Netflix Prize* [11] contest for movie recommendation performance. It equals to the square root of the MSE metric.

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}$$

3.3.3 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is another widely used metric in CF research, which measures the average of the absolute difference between the predicted value and the true value.

$$\text{MAE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

Where $\hat{R}$ is the total number of ratings over all users, r_{ui} is the predicted rating for user u on item i, and $\hat{r}_{ui}$ is the actual rating.

The difference between MAE and MSE/RMSE is that MSE/RMSE heavily emphasizes large errors. For example, given a test set with four hidden items, system A makes an error of 2 on three ratings and no error on the fourth, system B makes error of 3 on one rating and no error on all three others. In this scenario, MSE/RMSE would prefer system A and MAE would prefer system B.

3.3.4 Fraction of Concordant Pairs (FCP)

Concordant pairs and discordant pairs compare two pairs of data points and to describe their relationship. To calculate this, the data are processed in ordinal. The procedure in calculating concordant and discordant pairs compares the classifications of two variables (e.g., user 1 and user 2) on the same two items (item *i* and item *j*). If their direction of classifications is the same, the pairs are concordant. For example, both user 1 and user 2 rate item *i* higher than item *j*. If the direction of the classification is different, the pair is discordant (Koren and Sill, 2013). For example, user 1 rates item *i* higher than item *j* but user 2 rates item *i* lower than item *j*.

Given a test set $\hat{R}$, we define the number of concordant pairs n_c^u for user u by counting those ranked correctly by rating predictor $\hat{r}_u$. We count the discordant pairs n_d^u for user u in a similar way:

$$n_c^u = \left| \left\{ (i,j) \middle| \hat{r}_{ui} > \hat{r}_{uj} \;\&\; r_{ui} > r_{uj} \right\} \right|$$

$$n_d^u = \left| \left\{ (i,j) \middle| \neg(\hat{r}_{ui} > \hat{r}_{uj} \;\&\; r_{ui} > r_{uj}) \right\} \right|$$

Summing over all users we define $n_c = \sum_u n_c^u$ and $n_d = \sum_u n_d^u$:

$$n_c = \sum\nolimits_u n_c^u$$

$$n_d = \sum\nolimits_u n_d^u$$

The final fraction of concordant pairs denoted as following:

$$FCP = \frac{n_c}{n_c + n_d}$$

4. Results and Discussion

We performed three baseline models (MinPredictor, MaxPredictor, and NormalPredictor), three Neighborhood-based models (*kNN* Basic, *kNN* with Means, and *kNN* with Z-Score) with three similarity metrics (Cosine, MSD, and Pearson), *Slope One* model, and two Matrix Factorization-based models (*SVD* and *SVD++)* on our dataset, and evaluated their performance by using four evaluation metrics (RMSE, MSE, MAE, and FCP). Their results are reported in Table 4.1, and graphs are Figure 4.1 - 4.4.

Table 4.1 Experiment results for all models

	RMSE	MSE	MAE	FCP
Min Predictor	0.7007	0.4909	0.4909	0.0000
Max Predictor	0.7135	0.5091	0.5091	0.0000
Normal Predictor	0.6137	0.3766	0.4980	0.4796
KNN Basic-userBased-Cosine	0.5095	0.2596	0.3711	0.6361
KNN with Means-userBased-Cosine	0.4142	0.1716	0.3047	0.6844
KNN with ZScore-userBased-Cosine	0.4120	0.1698	0.3001	0.6843
KNN Basic-itemBased-Cosine	0.4790	0.2294	0.3629	0.5863
KNN with Means-itemBased-Cosine	0.4830	0.2333	0.3434	0.6687
KNN with ZScore-itemBased-Cosine	0.4842	0.2345	0.3404	0.6675
KNN Basic -userBased-MSD	0.5401	0.2917	0.4041	0.6261
KNN with Means-userBased-MSD	0.4324	0.1870	0.3207	0.6640
KNN with ZScore-userBased-MSD	0.4298	0.1847	0.3150	0.6639
KNN Basic-itemBased-MSD	0.4935	0.2436	0.3652	0.6414
KNN with Means-itemBased-MSD	0.4545	0.2066	0.3324	0.6824
KNN with ZScore-itemBased-MSD	0.4514	0.2037	0.3256	0.6890

KNN Basic-userBased-Pearson	0.4946	0.2447	0.3941	0.5244
KNN with Means-userBased-Pearson	0.4165	0.1735	0.3234	0.5607
KNN with ZScore-userBased-Pearson	0.4153	0.1725	0.3210	0.5611
KNN Basic-itemBased-Pearson	0.4890	0.2391	0.3619	0.5831
KNN with Means-itemBased-Pearson	0.4811	0.2315	0.3422	0.6578
KNN with ZScore-itemBased-Pearson	0.4791	0.2295	0.3401	0.6765
SlopeOne	0.4674	0.2185	0.3569	0.6142
SVD	0.3917	0.1534	0.3201	0.7002
SVD++	0.3859	0.1489	0.3104	0.6978

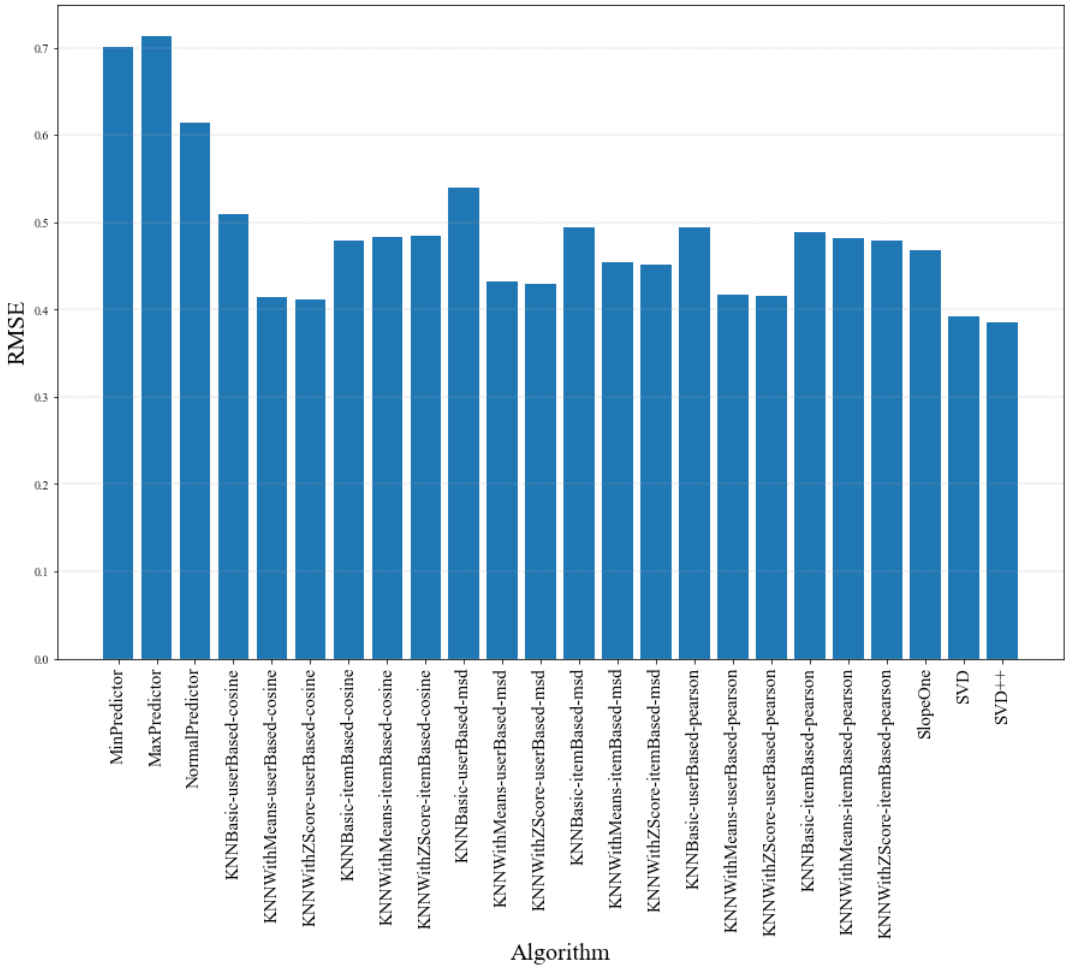

Figure 4.1 Comparing all models on RMSE

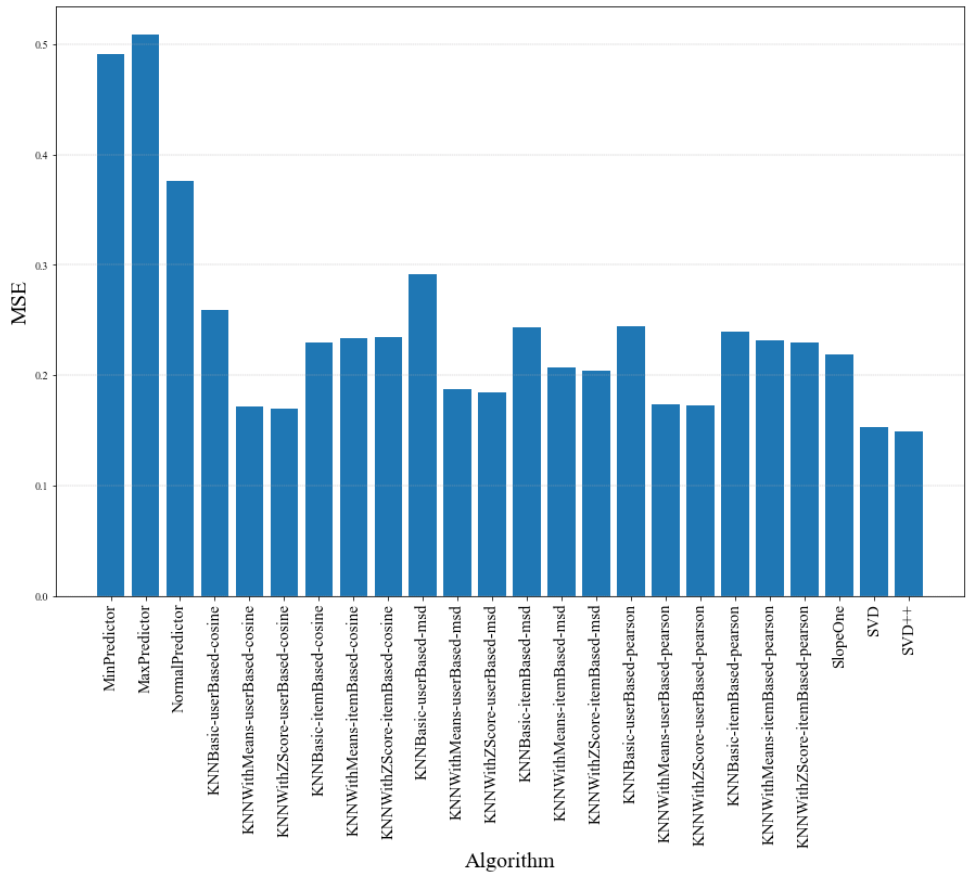

Figure 4.2 Comparing all models on MSE

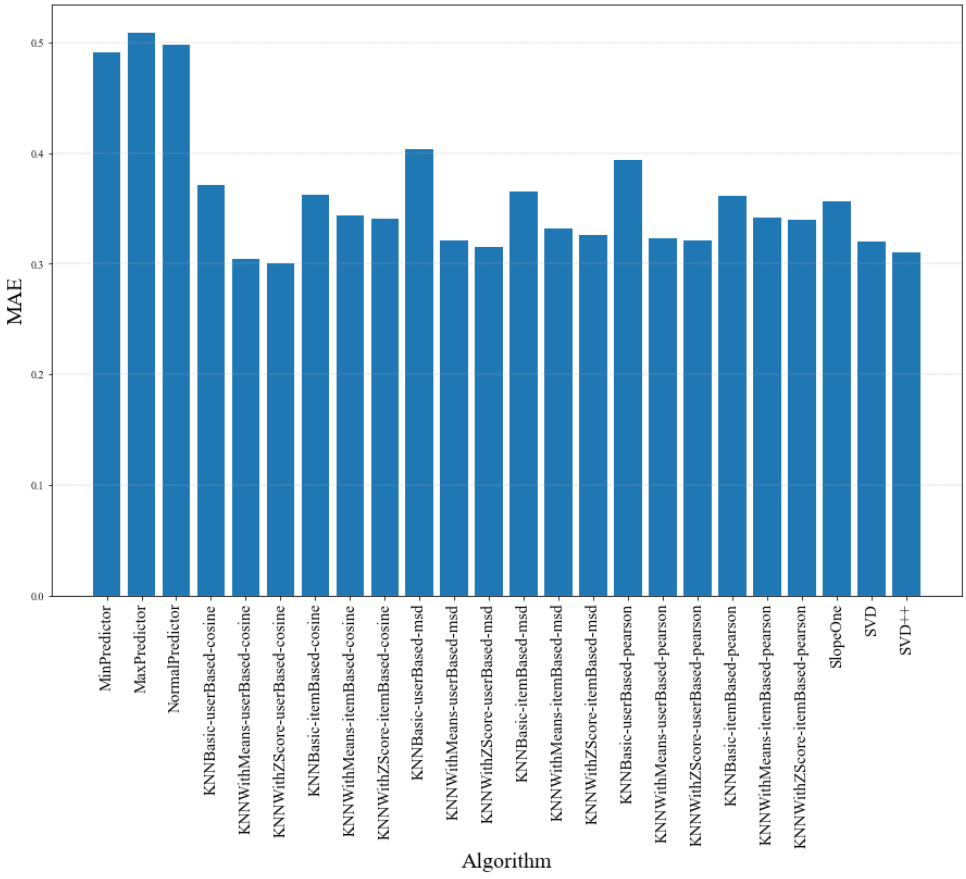

Figure 4.3 Comparing all models on MAE

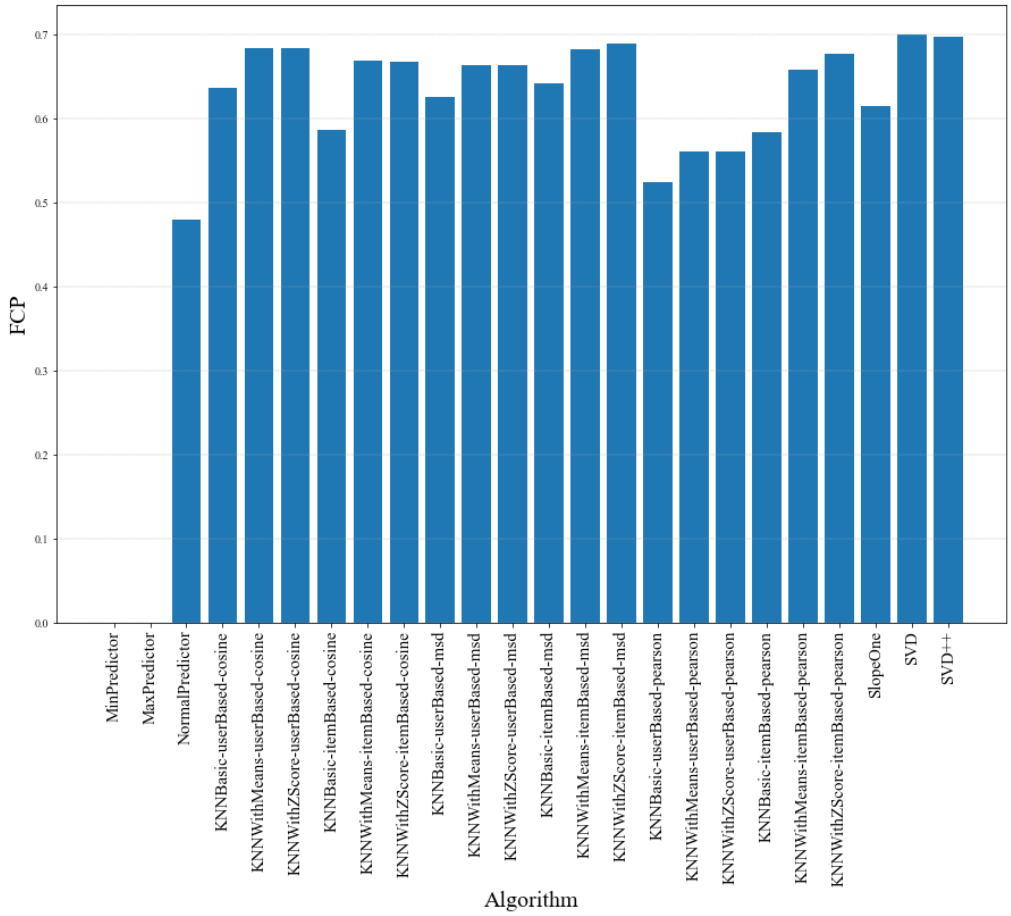

Figure 4.4 Comparing all models on FCP

4.1 Baselines

The results of RMSE, MSE, and MAE represent the gaps between the predicted values and the true values, the smaller their value, the better the predictions. While the result of FCP represent the fraction of correct predictions in total predictions, the higher this value, the better the predictions.

The results of our three baseline predictors proved that all other CF algorithms we tested are all working on our Chinese music dataset. Our baseline predictor Min Predictor always predict 0, and Max Predictor always predict 1. For every prediction, it's true value is either 1 or 0, so Min Predictor and Max Predictor always have 50 percent chance to reach the correct predictions, which is equivalent to random guessing. The MSE, and MAE results of Min

Predictor and Max Predictor are around 0.5, which fit our expectation. The RMSE is just the square root of the MSE. Since all pairs of value in Min Predictor and Max Predictor are same and cannot be treated as ordinal, so their FCP are 0. Since our Normal predictor predicts a random rating based on the distribution of the training set using maximum likelihood estimation, which is a more supplicated baseline than Min Predictor and Max Predictor, it is not surprised to see it outperform the other two on RMSE, MSE, and FCP metrics. From the MAE indicator, all the three baselines are around 0.5. We can intuitively understand that whether it is 0 or 1, the baseline models predict 0.5 on average, which means none of the three baselines did any predictive jobs in the score predictions of 0 and 1.

The MAE of all other recommendation algorithms is around 0.3 to 0.4. Take the best one, user-based *kNN* with Z-Score using Cosine similarity, as the example, an intuitive understanding is that, on average, the model prediction value is 0.3001 off from 0, and 0.6999 off from 1. For all evolution metrics, whether the algorithm is based on *kNN*, slope one, or matrix factorization, they significantly outperform our baselines, which indicates that those models all work on our data set, showing their predictive ability on the music recommendation task.

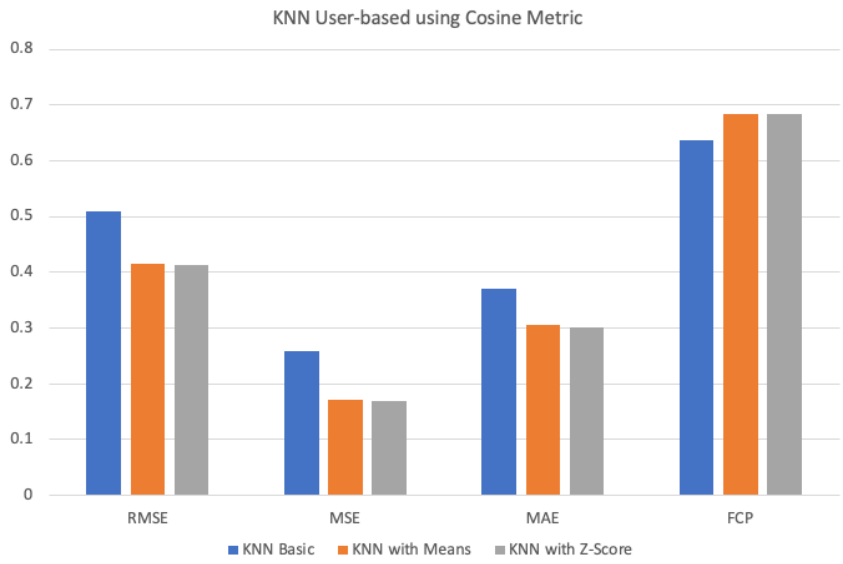

Figure 4.5 Comparing user-based *kNN* models on Cosine

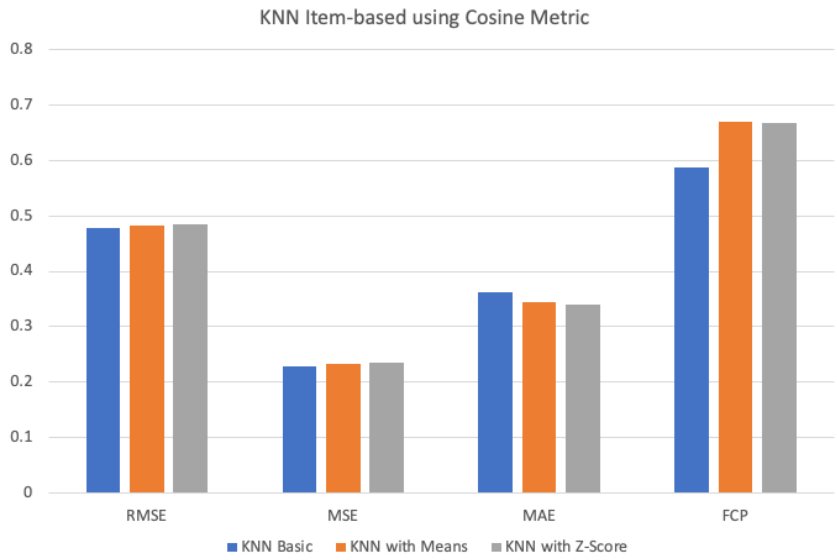

Figure 4.6 Comparing item-based *kNN* models on Cosine

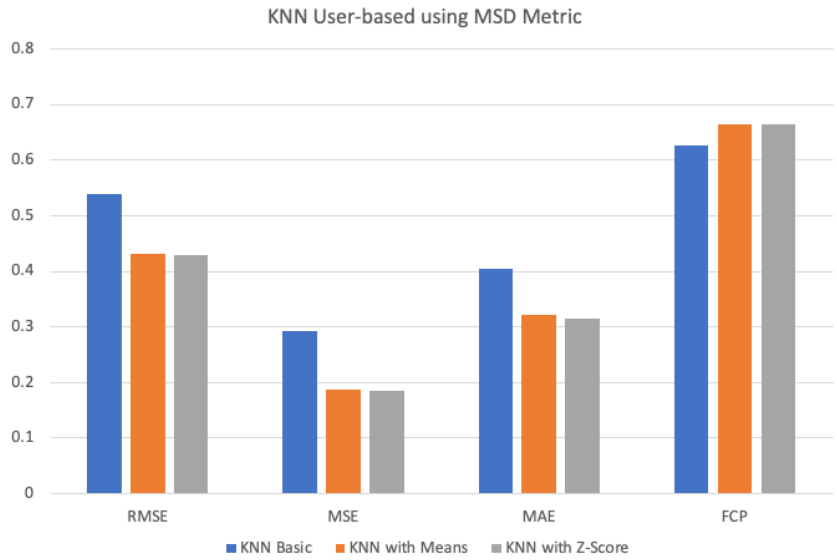

Figure 4.7 Comparing user-based *kNN* models on MSD

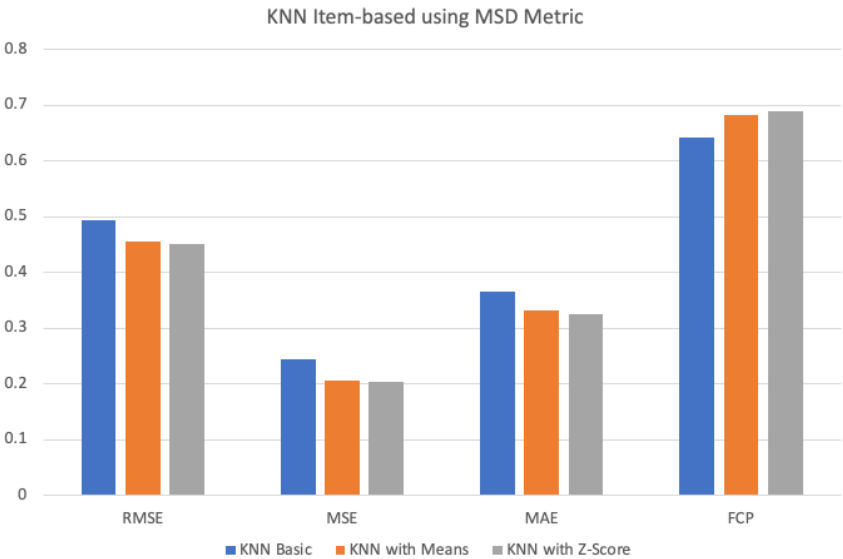

Figure 4.8 Comparing item-based *kNN* models on MSD

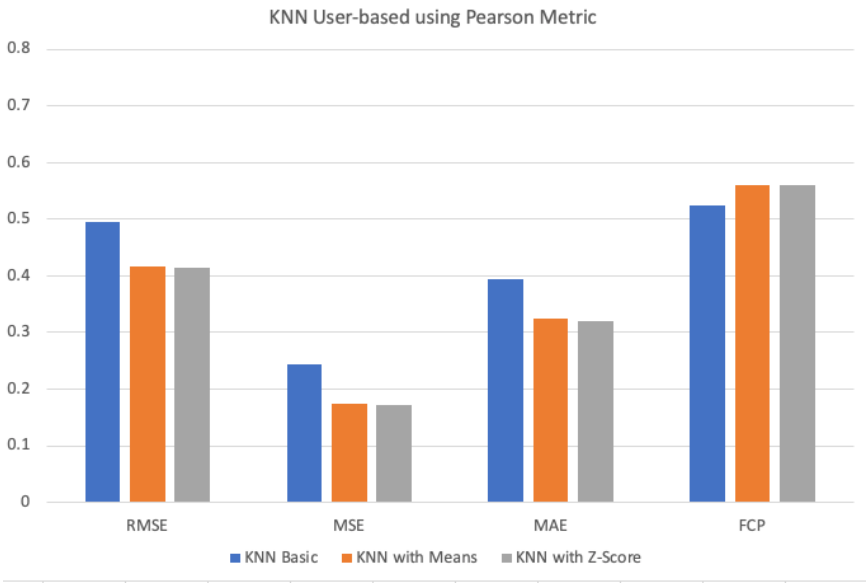

Figure 4.9 Comparing user-based *kNN* models on Pearson

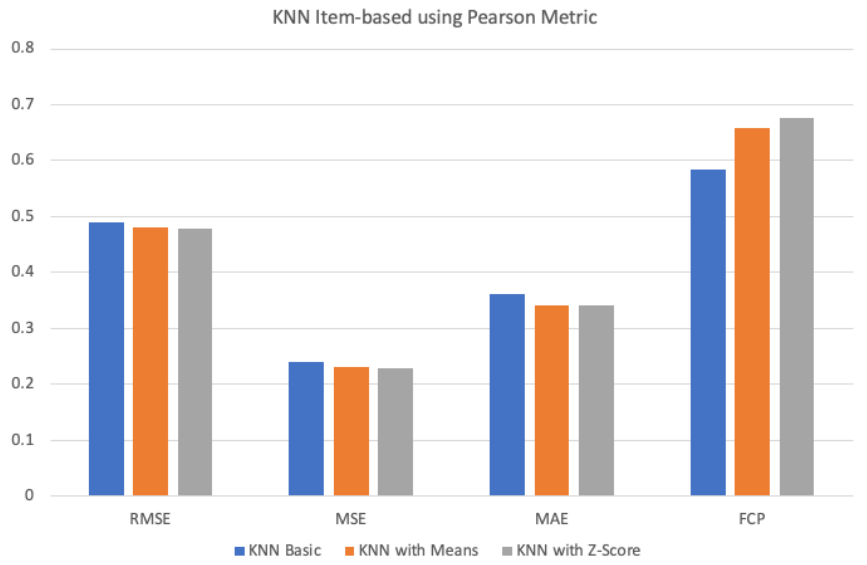

Figure 4.10 Comparing item-based *kNN* models on Pearson

4.2 *kNN*-based Algorithms

In the *kNN*-based recommendation algorithm, we chose three different similarity metrics (Cosine, MSD, and Pearson) in their calculation. As we can see on Figure 4.5 - 4.10, although their final results have slight differences, the trends between different models are consistent, which shows the robustness of each model in our music dataset.

For the two normalization methods, we found that almost all the *kNN* algorithm considering Means or Z-Score got more or less improved in their accuracy. This shows the effectiveness of such simple step in CF models. Both Means and Z-Score use mean centering to eliminate deviations caused by the different perceptions of the average rating, and Z-Score also considers the spread in each individual scales by considering their standard deviation. By comparing the two methods, we did not see an obvious difference. Z-Score improved a slightly accuracy on these models, usually around 0.01 to 0.03. The most significant one is FCP of item-based *kNN* using Pearson metric, has an improvement of 0.0623.

However, it can be seen from the charts that the user-based *kNN* algorithms experienced more influence than the item-based *kNN* by normalization. In all user-based *kNN* algorithms, *kNN* with normalization got significantly improve compare to *kNN* basic. This is because the ratings are highly influenced by the scale of individual users. Individuals have different standards and habits during rating. Some individuals are “kind-hearted” or have lower standards when rating and tend to rate high, while others may be “cold-hearted” or have higher standards and tend to give relatively low ratings. Normalization compensates for users' behavior by adjusting the rating scale to be comparable or on the same level as other users' ratings. This also shows that it is effective and necessary to do data normalization for user-based *kNN* methods.

To compare the results of user-based *kNN* and item-based *kNN*, with normalization, we can see user-based algorithms perform a bit better than item-based methods. The reason here may because we only took those playlists with more than one hundred subscribers. More subscribers may also indicate the playlist contain more popular songs. If a considerable number of songs in these playlists have overlap, those users are similar in the first place. So that user-based model can perform better.

Another reason for this may derive from the ratio between the number of users and items in our dataset. User-based methods obtain *k* similar users by comparing the ratings made by users on same items. Item-based methods obtain *k* similar items by comparing ratings made by the same users on items. Since we have 1076 users and 49774 items, the number of items is much larger than that of users. So that each user could compare lots of items to find their precise neighbors, but each item has much smaller data to consider their neighbors. However, the ratings in real-life situations are normally skewed. The majority of ratings are given to a small amount of items by lots of users, which means the number of users is much larger than that of items, just like platforms such as Amazon and Netflix. We can generalize that in the cases where the number of users is much greater than the number of items, item-based methods could produce more accurate recommendations. In the contrary, when we want to build recommender systems for platforms where the number of users less than the number of items, the user-based methods might perform better. For example, in research paper platforms, it may have tons of papers but only thousands of scholar users. In this case, the user-based methods could be implemented.

4.3 Slope One Algorithm

In our experiment, Slope One algorithm reaches medium-level results. In RMSE and MSE metrics, Slope One did worse than user-based *kNN* with normalization and two matric factorization-based (*SVD* and *SVD++*) algorithms, but outperform item-based *kNN* methods (whether with normalization or not). In MAE and FCP metrics, Slope One only did better than basic *kNN*. In short, Slope One has reasonably accurate results on our data set, despite its nature of simplicity. It expects little information from users, and its process time is much shorter than

other sophisticated algorithms. It might be used as a supplement to other sophisticated algorithms.

4.4 Matrix Factorization-based Algorithms

SVD and *SVD++* performed very well on our four similarity metrics. They outperform all other algorithms to a great extent. Especially for *SVD++*, it is overall the best algorithm on our dataset. *SVD++* outperforms *SVD* on RMSE, MSE, and MAE, but a slightly worse than *SVD* on the FCP metric. Its MAE reached 0.3104, which means, on average, its predictions deviate 31.04 % from the true value. The reason for *SVD++* performs better than *SVD*, mainly because *SVD++* takes users' implicit feedback as a second latent factors into considerations. For our dataset, the rating information was generated from user's collection behavior, and for their disliked songs, we artificially generate them by sampling the songs outside of the collection. All these ratings can be considered as their implicit feedback. So *SVD++* is very helpful in our cases.

Our results also show that matrix factorization-based methods in most cases have better predictive ability than *kNN*-based methods. Factorization-based and neighborhood-based methods developed upon different basic ideas and principles, which led to different prediction rules. *kNN* models try to connect users to new items by following the chains of user-item relationships. Such relationships represent their preference between the users and items. Both user-based and item-based models operate following the chains of user-item relationships, which can be seen as a local perspective. Factorization-based models try to observe ratings as the result of a number of latent factors of user characteristics (for our case, e.g., musical preference or taste, users' personality) and item characteristics (e.g., genre, instrumentation,

ect.). For each user, factorization-based models create a profile for them and make personalized recommendations, which can be regard as a global perspective.

However, we can see on the MAE metric, all user-based *kNN* models with normalization reached comparable results with matrix factorization-based models. The best result is actually reached by user-based *kNN* models using cosine similarity with Z-Score, even 1.03% better than *SVD++*. But on RMSE and MSE metrics, matrix factorization-based still way better than *kNN* models. The reason here is related to the difference between the evaluation principles of RMSE/MSE and MAE. Comparatively, (R)MSE penalizes large gaps more harshly than MAE. It shows that in our dataset, there are some data points that would easily make our algorithms' prediction largely away from the real value. *SVD* and *SVD++* have better performance when facing these outliers, so their (R)MSE results are better. However, the penalty for the outlier in MAE is smaller, so that the advantages of *SVD* and *SVD++* are not reflected. Using the *kNN* algorithm with normalization, the rating scale can be well controlled in a smaller range, so their MAE can also get good results. At the same time, in the parameter estimation of *SVD* and *SVD++*, the loss function uses regularized squared error, so the model will update the parameters in the direction of decreasing squared error, so the model tends to eventually converge to a smaller (R)MSE. Thus, a better result also depends on the error measure we use. If we want an unbiased forecast, use the (R)MSE would be more suitable. If we want the median of the future distribution, use the MAE could get a better result.

4.5 Limitations

Our experiment has some limitations. The most obvious one is in the process we develop our dataset. We created the binary rating matrix by giving "1" (stands for "rated and like") and "0" (stands for "dislike") to the songs in the collection and outside the collection arbitrarily. We

assume users have listened to all the songs in our dataset, but this is ideal and may not be true. For the songs added to the collection, we gave them 1, which is not disputed. However, there are many reasons why a song is not added to the collection, including the user indeed dislike it, have not listened to it, or like it but forgot to add it to the collection, etc. For this part of user context information, there may be better ways to deal with it, but based on our knowledge, the best way we can do it is to dualize them. Fortunately, our data proved to be valid through experiments. If we take a smarter approach, we may get better results.

Another limitation of the experiment is that we filtered out many playlists with data sparsity issues by only keeping playlists with the *subscribedCount* > 100. How to deal with this part of the user's information has always been a difficult problem in the CF algorithm, also known as the cold-start problem. When we have users who have little information to use, how do we recommend items to them? Our experiments choose to ignore this part of information, since these data will cause interference to other playlists that do not have data sparsity issues. We will further discuss this problem in future work.

Another point we want to bring out is that, for the evolution metrics we use, they only show the overall quality of our algorithm in predictions. For the recommendations for a single user, we cannot see their quality through these metrics. Because each of our algorithms actually scores each item for each user, these ratings actually form a ranking, so we may be able to use the idea of discounted cumulative gain metric to make recommendations for each user. Since each of our ratings has different errors, we can give different errors a different relevance score. For example, we give those errors less than 0.1 a score of 4, error less than 0.2 a score of 3, and error less than 0.3 a score of 1, etc. We can also add normalization to make it a Normalized Discounted Cumulative Gain. In this way, we can get the recommendation quality of these algorithms for a single user.

5. Conclusion and Future Works

In this thesis, we explored the recommendation task in Music Information Retrieval area. By reviewing the three state-of-the-art approaches (music content-based, music text-based, and user context-based) towards the task, we narrow down our focus to the user context-based direction. We found that there are few and old available datasets in this direction, and there is no public dataset specially developed for Chinese music. So, we developed a new dataset from the mainstream Chinese music streaming platform NetEase Cloud Music. Moreover, although various algorithms have been born in the research of recommendation systems, in the field of music recommendation, there is no systematic comparison of various collaborative filtering algorithms and testing their effectiveness. So, we compared the performance of a series of Memory-based and Model-based collaborative filtering algorithms on our dataset. Our experimental results prove that these CF algorithms aiming at users' information are effective on our dataset, and they have the predictive ability of music recommendation task on Chinese music data. In general, model-based algorithms perform better than Neighborhood-based algorithms. Within them, the *SVD++* from Matrix Factorization-based algorithms work best for this type of task.

For future works, according to our experiments, we believe that the effectiveness of music recommendation task can be improved in terms of improving the reliability of the data source and improving the accuracy of the algorithm. In our experiment, the user rating data we use comes from their collection behavior, which is implicit feedbacks. The lack of obvious explicit feedback is due to the user's habit of listening to music. If the user does not like a piece of music, their usual behavior is to skip it, and this part of the data cannot be obtained by crawling. This is different from user ratings commonly used in movie recommendations. We

believe that in future works, for user ratings, we can consider the total playing times and playback times as an indicator to build their rating matrix, which may be a more reliable user behavior data source.

Another takeaway is that the distinctions consider neighborhood models as “memory-based”, while taking matrix factorization as “model-based” is not always appropriate. When we use sophisticated neighborhood models that take normalization methods into account, their presentations are delicate enough to be considered as a "model", and some of them performed as well as model-based methods, even outperform model-based in some evaluation metrics. Moreover, the opposite direction is also true. Those matrix factorization models with better performance also following memory-based idea, since they sum and calculate the overall memory stored when making predictions. Therefore, when we want to achieve higher accuracy, we do not have to be limited to which memory-based or model-based method, but combine their ideas to make a hybrid method. Memory-based model is like a local perspective, by considering their similar neighbors; while model-based is like a global perspective, by using the equation to go through all the data. Actually, our *kNN* models with normalization is just like add a global perspective by taking the mean and Z-Score of all rating to *kNN*’s local perspective. *SVD++* is just like add a local perspective by taking a set of item rating as a new latent factor, in which the idea here is very similar to item-based *kNN*. A potential solution to achieve a higher accuracy is by applying a more limited and localized neighborhood models, where the number of *k* is small. The small *k* value might not be the best way to construct a neighborhood model alone, but it makes the neighborhood model possible to add a local perspective to the factorization model's global perspective. Generally, we can even think of combining the features of music content-based, music context-based and user context-based methods, to make a multimodal recommender system. Although the previous two methods are

not discovered in this thesis, I would explore them in my future research and trying to connect them.

Lastly, when we think about what kind of music recommendations can satisfy users, music similarity will be the first indicator that pops out in our minds: users like the music they are familiar with. On our data set, our best algorithm yields about 30% error, which is a relatively satisfactory result in real-life situations. However, it seems not enough. As human beings, we also want to explore some new stuff that might interest us. If users always get similar music recommendations, they will eventually get bored. So, besides only concentrate on the errors in music similarities, we can add another consideration: freshness and novelty. We hope that through the behavior of users, we can also predict the types of fresh music that they might like and they have not been exposed to before, but it cannot be completely irrelevant. Maybe some new methods like *SVD*, considering those ratings in lower-dimensional latent space could be developed to describe these user preferences. By exploring more needs of users, we could achieve a higher level of music recommendation in the future.

Bibliography

1. https://jmcauley.ucsd.edu/data/amazon/
2. https://grouplens.org/datasets/movielens/
3. https://www.netflixprize.com/index.html/
4. http://millionsongdataset.com/
5. https://research.atspotify.com/datasets/
6. https://webscope.sandbox.yahoo.com/catalog.php?datatype=r/
7. https://www.last.fm/
8. https://www.allmusic.com/
9. https://music.163.com/
10. https://www.scraperapi.com/
11. https://www.netflixprize.com/
12. Ricci, F., Rokach, L., and Shapira, B. (2015). *Recommender Systems Handbook*. Springer.
13. Pachet, F. (2005). Knowledge management and musical metadata. *Idea Group*, 12.
14. 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*, 210–217.
15. Cohen, W. W., and Fan, W. (2000). Web-collaborative filtering: recommending music by crawling the Web. *Computer Networks, 33*(1), 685–698.
16. Schedl, M., Knees, P., McFee, B., Bogdanov, D., and Kaminskas, M. (2015). Music Recommender Systems. In Ricci, F., Rokach, L., and Shapira, B. (Eds.), *Recommender Systems Handbook* (pp. 453–492). Springer.
17. Whitman, B., and Lawrence, S. (2002). Inferring descriptions and similarity for music from community metadata. *In Proceedings of the International Computer Music Conference (ICMC)*.

18. Geleijnse, G., Schedl, M., and Knees, P. (2007). The Quest for Ground Truth in Musical Artist Tagging in the Social Web Era. *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 525–530.

19. Mahedero, J. P. G., MartÍnez, Á., Cano, P., Koppenberger, M., and Gouyon, F. (2005). Natural language processing of lyrics. *In Proceedings of the 13th Annual ACM International Conference on Multimedia*, 475–478.

20. Kleedorfer, F., Knees, P., and Pohle, T. (2008). Oh Oh Oh Whoah! Towards Automatic Topic Detection in Song Lyrics. *In Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 287–292.

21. Mayer, R., Neumayer, R., and Rauber, A. (2008). Combination of audio and lyrics features for genre classification in digital audio collections. *In Proceeding of the 16th ACM international conference on Multimedia*.

22. Hu, X., Downie, J. S., and Ehmann, A. F. (2009). Lyric text mining in music mood classification. American Music, 183(5,049), 2–209.

23. Pampalk, E., Rauber, A., and Merkl, D. (2002a). Content-based Organization and Visualization of Music Archives. *In Proceedings of the 10th ACM International Conference on Multimedia*, 570–579.

24. Dixon, S., Pampalk, E., and Widmer, G. (2003). Classification of Dance Music by Periodicity Patterns. *In Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR)*, 159–166.

25. Dixon, S., Gouyon, F., and Widmer, G. (2004). Towards Characterisation of Music via Rhythmic Patterns. *In Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, 509–516.

26. Gouyon, F., Dixon, S., Pampalk, E., and Widmer, G. (2004). Evaluating Rhythmic Descriptors for Musical Genre Classification. *In Proceedings of the 25th AES International Conference.*

27. Burred, J.J., and Lerch, A. (2003). A Hierarchical Approach to Automatic Musical Genre Classification. *In Proceedings of the 6th International Conference on Digital Audio Effects (DAFx).*

28. Foote, J. T. (1997). Content-Based Retrieval of Music and Audio. *In Proceedings of SPIE Multimedia Storage and Archiving Systems II*, 3229, 138–147.

29. Logan, B. (2000). Mel Frequency Cepstral Coefficients for Music Modeling. *In Proceedings of the International Symposium on Music Information Retrieval*.

30. Baccigalupo, C., Plaza, E., and Donaldson, J. 2008. Uncovering affinity of artists to multiple genres from social behaviour data. *In Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR).*

31. Logan, B. and Salomon, A. (2001). A Music Similarity Function Based on Signal Analysis. *In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*.

32. Aucouturier, J.-J., and Pachet, F. (2004). Improving Timbre Similarity: How High is the Sky? *Journal of Negative Results in Speech and Audio Sciences*, *1*(1).

33. Aucouturier, J.-J., Pachet, F., and Sandler, M. (2005). "The Way It Sounds": Timbre Models for Analysis and Retrieval of Music Signals. *IEEE Transactions on Multimedia*, 7(6):1028–1035.

34. Aucouturier, J.-J., Pachet, F., Roy, P., and Beurivé, A. (2007). Signal + Context = Better Classification. *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR).*

35. Mandel, M. I. and Ellis, D. P. (2005). Song-Level Features and Support Vector Machines for Music Classification. *In Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR).*

36. Pohle, T. and Schnitzer, D. (2007). Striving for an Improved Audio Similarity Measure. *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR).*

37. Pachet, F., Westerman, G., and Laigre, D. (2001). Musical Data Mining for Electronic Music Distribution. *In Proceedings of the 1st International Conference on Web Delivering of Music.*

38. Koren, Y., and Sill, J. (2013). Collaborative filtering on ordinal user feedback. *In Proceedings of the 23th International Joint Conference on Artificial Intelligence*.

39. Casey, M.A., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., and Slaney, M. (2008). Content-Based Music Information Retrieval: Current Directions and Future Challenges. *In Proceedings of the IEEE 96*, 668–696.

40. Li, D., Sethi, I.K., Dimitrova, and N., and McGee, T. (2001) Classification of General Audio Data for Content-based Retrieval. *Pattern Recognition Letters 22*(5), 533–544

41. Scheirer, E., and Slaney, M. (1997). Construction and Evaluation of a Robust Multifeature Speech/Music Discriminator. *In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* 1331–1334.

42. Gouyon, F., Pachet, F., and Delerue, O. (2000). On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds. *In Proceedings of the COSTG6 Conference on Digital Audio Effects (DAFx).*

43. Pohle, T. (2009). Automatic Characterization of Music for Intuitive Retrieval. PhD thesis, Johannes Kepler University Linz, Linz, Austria.

44. Vikram, D. and Shashi, M. (2017). A framework for pattern-based melody matching for Content Based Music Information Retrieval. *In Proceedings of the Intelligent Systems Conference (IntelliSys).*

45. Knees, P. and Widmer, G. (2008). Searching for Music Using Natural Language Queries and Relevance Feedback. *In Proceedings of the Lecture Notes in Computer Science.* 4918, 109–121.

46. Breese, J. S., Heckerman, D., and Kadie, C. (2013). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *In Proceedings of the 14th Annual Conf. on Uncertainty in Artificial Intelligence*, pp. 43–52.

47. Billsus, D., Pazzani, M. J., and Others. (1998). Learning collaborative information filters. *In Proceedings of the International Conference on Machine Learning (ICML)*, 46–54.

48. Deshpande, M., and Karypis, G. (2004). Item-based top-N recommendation algorithms. *ACM Transactions on Information and System Security*, 22(1), 143–177.

49. Pandey, A., Bharati Vidyapeeth's College of Engineering, Information Technology, New Delhi, 110063, India, & Jain, A. (2017). Comparative analysis of KNN algorithm using various normalization techniques. *International Journal of Computer Network and Information Security*, 9(11), 36–42.

50. Mesnage, C. S., Rafiq, A., Dixon, S., and Brixtel, R. P. (2011). Music discovery with social networks. *In Proceedings of the 2nd Workshop on Music Recommendation and Discovery (WOMRAD)*. 7–12.

51. Desrosiers, C., and Karypis, G. (2011). A Comprehensive Survey of Neighborhood-based Recommendation Methods. In Ricci, F., Rokach, L., and Shapira, B. (Eds.), *Recommender Systems Handbook* (pp. 107–144). Springer.

52. Celma Herrada, Ò. (2009). Music recommendation and discovery in the long tail. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain.

53. Slaney, M., and White, W. (2007). Similarity Based on Rating Data. *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 479–484.

54. Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. (1997). GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3), 77–87.

55. Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), 76–80.

56. Nanopoulos, A., Radovanović, M., and Ivanović, M. (2009). How does high dimensionality affect collaborative filtering? *In Proceedings of the Third ACM Conference on Recommender Systems*, 293–296.

57. Lemire, D., and Maclachlan, A. (2005). Slope One Predictors for Online Rating-Based Collaborative Filtering. *In Proceedings of the 2005 SIAM International Conference on Data Mining* (pp. 471–475).

58. Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information and System Security*, 22(1), 89–115.

59. Sarwar, B. M., Karypis, G., Konstan, J., and Riedl, J. (2002). Incremental SVD-based algorithms for highly scaleable recommender systems. *In Proceedings of the Fifth International Conference on Computer and Information Technology*, 345–354.

60. Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *IEEE Computer*, 42(8), 30–37.

61. Drineas, P., Kerenidis, I., and Raghavan, P. (2002). Competitive recommendation systems. *In Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 82–90.

62. Logan, B. (2004). Music Recommendation from Song Sets. *In Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, 425–428.

63. Dror, G., Koenigstein, N., Koren, Y., and Weimer, M. (2012). The Yahoo! music dataset and KDD-cup'11. *In Proceedings of the KDD Cup 2011*, 3–18.

64. Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. *In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 426–434.

65. Hug, N. (2020). Surprise: A Python library for recommender systems. *Journal of Open Source Software*, 5(52), 2174.

Appendix

Appendix A: A sample JSON format of each playlist


```
{
  "result": {
      "coverImgUrl": "http://p1.music.126.net/m9cBIL-D3or61yMh9OPdqg==/82463372088816.jpg
    ",
      "ordered": true,
      "anonimous": false,
      "creator": {
            "followed": false,
            "remarkName": null,
            "expertTags": null,
            "userId": 16017674,
            "authority": 0,
            "userType": 0,
            "gender": 1,
            "backgroundImgId": 2002210674180202,
            "city": 1004400,
            "mutual": false,
            "avatarUrl": "http://p1.music.126.net/n4CohDjvn9d8tEqjNY5vBQ==/79142846969353
    66.jpg",
            "avatarImgIdStr": "7914284696935366",
            "detailDescription": "",
            "province": 1000000,
            "description": "",
             "birthday": 631170000000,
             "nickname": "Michaellincolic",
```

```
            "vipType": 10,

            "avatarImgId": 7914284696935366,

            "defaultAvatar": false,

            "djStatus": 0,

            "accountStatus": 0,

            "backgroundImgIdStr": "2002210674180202",

            "backgroundUrl": "http://p1.music.126.net/pmHS4fcQtcNEGewNb5HRhg==/2002210674180202.jpg",

            "signature": "乡愁是一枚小小的邮票，而我是乡愁。",

            "authStatus": 0

             },

      "trackUpdateTime": 1493994741396,

      "updateTime": 1475209156574,

      "commentCount": 2,

      "artists": null,

      "newImported": false,

      "commentThreadId": "A_PL_0_58631200",

      "subscribed": false,

      "privacy": 0,

      "id": 58631200,

      "trackCount": 182,

      "specialType": 0,

      "status": 0,

      "description": "在 80 后记忆里，那时候的大街小巷，还没有网络神曲，那时候我们用随身听一个个传着听的经典流行歌曲，是时代的烙印",

      "subscribedCount": 160,

       "tags": [

              "怀旧",

              "80 后",

              "华语"

           ],
```

```
            "coverImgId": 82463372088816,
            "tracks": [{Track 1},{Track 2}, {Track 3},...],
            "highQuality": false,
            "subscribers": [],
            "playCount": 5479,
            "trackNumberUpdateTime": 1475209141421,
            "createTime": 1426391701062,
            "name": "音乐的黄金年代-流行歌曲的经典记忆",
            "cloudTrackCount": 0,
            "shareCount": 1,
            "adType": 0,
            "totalDuration": 0
        }
}
```

Appendix B: A sample JSON format of each song


```
{
   "bMusic": {
              "name": null,
              "extension": "mp3",
              "volumeDelta": -0.31,
              "sr": 44100,
              "dfsId": 7959364674930721,
              "playTime": 297000,
              "bitrate": 96000,
              "id": 98780031,
              "size": 3571230
              },
    "hearTime": 0,
    "mvid": 509095,
    "hMusic": {
               "name": null,
               "extension": "mp3",
               "volumeDelta": -0.7,
               "sr": 44100,
               "dfsId": 7845015465254314,
               "playTime": 297000,
               "bitrate": 320000,
               "id": 98780029,
               "size": 11903663
               },
     "disc": "",
     "artists": [
                 {
```

```
                "img1v1Url": "http://p1.music.126.net/6y-UleORITEDbvrOLV0Q8A==/5639395138885805.jpg",
                "name": "周杰伦",
                "briefDesc": "",
                "albumSize": 0,
                "img1v1Id": 0,
                "musicSize": 0,
                "alias": [],
                "picId": 0,
                "picUrl": "http://p1.music.126.net/6y-UleORITEDbvrOLV0Q8A==/5639395138885805.jpg",
                "trans": "",
                "id": 6452
                 }
               ],
      "duration": 297000,
      "id": 186011,
      "album": {
                 "status": 1,
                 "blurPicUrl": "http://p1.music.126.net/fvq0SpfXzZKgo8drc2KjOw==/51677046517811.jpg",
                 "copyrightId": 1007,
                 "name": "寻找周杰伦",
                 "companyId": 0,
                 "description": "",
                 "pic": 51677046517811,
                 "commentThreadId": "R_AL_3_18904",
                 "publishTime": 1067616000000,
                 "briefDesc": "",
                 "company": "阿尔发音乐",
                 "picId": 51677046517811,
                 "alias": [],
```

```
                "picUrl": "http://p1.music.126.net/fvq0SpfXzZKgo8drc2KjOw==/516770465
17811.jpg",

       "artists": [

                  {

                 "img1v1Url": "http://p1.music.126.net/6y-UleORITEDbvrOLV0Q8A==/56393
95138885805.jpg",

                 "name": "周杰伦",

                 "briefDesc": "",

                 "albumSize": 0,

                 "img1v1Id": 0,

                 "musicSize": 0,

                 "alias": [],

                 "picId": 0,

                 "picUrl": "http://p1.music.126.net/6y-UleORITEDbvrOLV0Q8A==/56393951
38885805.jpg",

                 "trans": "",

                 "id": 6452

                  }

                 ],

                  "songs": [],

                  "artist": {

                      "img1v1Url": "http://p1.music.126.net/6y-UleORITEDbvrOLV0Q8A==/
5639395138885805.jpg",

                      "name": "",

                      "briefDesc": "",

                      "albumSize": 0,

                      "img1v1Id": 0,

                      "musicSize": 0,

                      "alias": [],

                      "picId": 0,

                      "picUrl": "http://p1.music.126.net/6y-UleORITEDbvrOLV0Q8A==/563
9395138885805.jpg",

                      "trans": "",
```

```
            "id": 0
        },
        "type": "EP/Single",
        "id": 18904,
        "tags": "",
        "size": 4
    },
    "fee": 8,
    "no": 2,
    "rtUrl": null,
    "ringtone": "600902000006889324",
    "rtUrls": [],
    "score": 100,
    "rurl": null,
    "status": 0,
    "ftype": 0,
    "mp3Url": "http://m2.music.126.net/svF95I0Y5HQf0i9y5qiTBw==/7959364674930721.mp3",
    "audition": null,
    "playedNum": 0,
    "commentThreadId": "R_SO_4_186011",
    "mMusic": {
     "name": null,
     "extension": "mp3",
     "volumeDelta": -0.28,
     "sr": 44100,
     "dfsId": 7979155884225087,
     "playTime": 297000,
     "bitrate": 160000,
     "id": 98780030,
     "size": 5951925
```

```
    },
    "lMusic": {
        "name": null,
        "extension": "mp3",
        "volumeDelta": -0.31,
        "sr": 44100,
        "dfsId": 7959364674930721,
        "playTime": 297000,
        "bitrate": 96000,
        "id": 98780031,
        "size": 3571230
    },
    "copyrightId": 1007,
    "name": "断了的弦",
    "rtype": 0,
    "crbt": "e93f07bd4132712eaeed426ef78e89ca",
    "popularity": 93,
    "dayPlays": 0,
    "alias": [],
    "copyFrom": "",
    "position": 2,
    "starred": false,
    "starredNum": 0
}
```