SANT'ANNA

MASTER THESIS

Applying Content-Boosted Collaborative Filtering On Spotify Dataset

Author:
Khoi Hoang NGUYEN

Supervisor:
Prof Anna Monreale

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

in the

Research Group Name
Management Department

November 6, 2017

Declaration of Authorship

I, Khoi Hoang NGUYEN, declare that this thesis titled, "Applying Content-Boosted Collaborative Filtering On Spotify Dataset" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Thanks to Google, Wiki, Stack Overflow for helping me with this monster"

Khoi Hoang Nguyen

Sant'Anna

Abstract

Faculty Name
Management Department

Master thesis

Applying Content-Boosted Collaborative Filtering On Spotify Dataset

by Khoi Hoang NGUYEN

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 A statement of the problem

Recommender system has been an interesting subject of research for long. The idea of recommender system began in early 1990s - with the popularization of the internet, to utilize the critique of millions of people online to help us acquire more useful and interesting content. The PARC Tapestry system [31] first introduced the novel idea of collaborative filtering technique in 1992, which instantaneously became a subject of interest for many other research groups and was employed in many systems, including the GroupLens system [70], the Ringo system at MIT [81], and the Bellcore Video Recommender [36]. As the domain gained more attractions, various approaches and techniques have been developed in the field, such as the materialization of content-based approach in 1997 [6], or the knowledge-based approach that is embedded in the FindMe recommender system [87].

Music recommender system is a branch, among others, of the recommender domain. The purpose of a music recommender is to suggest songs that user might like amid a collection of millions of sound track. Many approaches have been proposed, including collaborative filtering to exploit the wisdom of the crowd, content-based approach to find similarity tracks, genre, or artist, and context-based approach to explore the relationship between context and listeners' behaviors.

Often, for content-based approach, acoustic features such as timbral are applied for genre classification. However, this approach looks at music recommendation as a purely signal processing problem, without taking into consider the intrinsic properties that humans perceive. Because acoustic features cannot capture semantic meaning of a track, an "energetic" song can be played next to a "nostalgic" one because of the similarity of the instrument, leading to a poor recommendation.

Semantic annotation is another promising trend in content-based approach. The idea of these approaches is to bridge the semantic gaps of acoustic features using machine learning techniques. However, this is a complex problem, as mapping between human annotation and acoustic features is not easy to define [5]. Many attempts have been made, such as the one that trains Gaussian mixture models of MFCCs for finding genre, moods, and instruments.

The Echo Nest [1], a music intelligence and data platform company, took another approach. Instead of exploring genre and moods similarity, the company extracts other kind of music features. Besides some well-known ones such as tempo and key, it also defines some more specialize ones, such as speechiness to denote the degree of spoken words, or danceability to describe how suitable a track is for dancing.

[1] http://the.echonest.com/

As Spotify [2], a digital music service, acquired The Echo Nest [84], the company attached Echo Nest's features into its own music library, creating a database of audio features for millions of track. As Spotify published the database [3], it creates chances for researchers and developers to develop an insight on the correlation between these variables and users' tastes, as well as to have a possibility to increase the quality of music recommendations.

However, Spotify does not provide user listening data. To overcome this difficulty, I tried to combine the famous last.fm [4] dataset with the one from Spotify. As I decide to work with album as the unit item, user data from last.fm is crawled on the album level. Of an album, all the track that exists as well as their acoustic features are then acquired from Spotify. The means of the features of the tracks of an album are taken as the representations of that album. More information on the construction of the dataset will be described in chapter 4.

In this thesis, I will compare three main approaches, including a pure collaborative filtering, a pure content-based, and a basic content-boosted collaborative filtering on the join dataset between last.fm and spotify datasets. The purpose is to see how well these parameters perform in pure content-based compare to pure collaborative filtering approach, as well as how the hybrid method resolve some of the pure collaborative filtering problem.

The following of the thesis is structured as follow: Chapter 2 describes fundamental approaches in recommender system and provides an overview of current music recommender's research. Chapter 3 explore the dataset as well as explains the detail of the system. Chapter 4 describes in detail the experiments, and the final chapter discusses the result of the research.

[2] https://www.spotify.com
[3] https://developer.spotify.com/web-api/get-audio-features/
[4] https://www.last.fm/

Chapter 2

Background information

In this chapter, I will firstly make a survey of different types of recommender systems; then I will narrow down the topic to music recommender systems. As music possess different features comparing to other kind of information such as movie and news, its recommender system also needs to be tailor to bring satisfaction to its listeners.

2.1 Overview of recommender systems

There are currently three main basic approach to recommendation, namely collaborative recommendation, content-based recommendation, and knowledge-based recommendation [39]. There are also an approach, called hybrid approaches, that tries to combine different recommendation together, in order to augment the strength and limit the drawback of each separate techniques. The description, as well as the advantages and disadvantages, of each recommendation techniques will be discussed as follow.

2.1.1 Collaborative recommendation

The main idea of collaborative recommendation approaches is to predict potential items that a user would like using past behavior information of other users. Pure collaborative algorithm take only user-item ratings as input and generate a prediction suggesting to what degree the user will like a certain item or a list of n recommended items as output.

There are two kind of ratings that can be used, namely implicit and explicit ratings. Explicit rating information can be collected by explicitly asking users to rate the item on a specific scale. Different scales are applied to different domain, as the quality of recommendation is different between these scales [24]. Ratings are then convert internally to numeric values in order for recommender systems to calculate the similarities. Implicit ratings, on the other hand, are knowledge collected based on the interaction between users and the systems. They can be in various forms with distinctive characteristics, such as information about item buying, book reading, music listening, or even user browsing behavior. As implicit ratings are observed behaviors, recommenders have to interpret whether the behaviors have positive or negative impacts toward the users. Even though the interpretation might be incorrect in some cases (e.g., a user might not like all the items that she bought), a massive amount of feedback would exclude these particular cases. In fact, Shafer et al [74] report that in some domains, user model using implicit information can outperform the one with explicit ratings.

There are many algorithms develop to exploit the rating matrix; However, in this review, I will just go into detail some main approaches that have been studied

carefully in the past and have been applied widely in the industry. These approaches include user-based approach, item-based approach, and an approach using matrix factorization/ latent factor models. Some other approaches, such as probabilistic approach, or slope one predictors, will not be mentioned here because of the scope of the thesis.

2.1.1.1 User-based nearest neighbor recommendation

User-based nearest neighbor recommendation is one of the earliest algorithm for this approach. The main idea of the algorithm is to identify other users who have similar preferences to a user; then for any item i that is unknown to the current user, a prediction is given based on the similar of the ratings of other users on item i.

2.1.1.1.1 Pearson's correlation coefficient

One common method to calculate the similar between users is Pearson's correlation coefficient. The general formula for the coefficient ρ [11] is:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

with $cov(X,Y)$ is the covariance between X and Y, and σ_X and σ_Y represent the standard deviation of X and Y.

Apply the formula to the case of collaborative filtering: let $\mathbf{U} = \{\mathbf{u_1}, \ldots, \mathbf{u_n}\}$ denote the set of users, $\mathbf{U} = \{\mathbf{p_1}, \ldots, \mathbf{p_n}\}$ for the set of items. The 2 sets form a $n \times m$ matrix of rating $r_{i,j}$ with $i \in 1 \ldots n, j \in 1 \ldots m$, with $r_{a,p}$ is the rating of user a on item p, and $\bar{r}_a$ denotes the average rating of user a. The similarity of user a and b $sim(a,b)$ is defined as follow:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum p \in P(r_{b,p} - \bar{r}_b)^2}}$$

The Pearson correlation coefficient takes value range from -1 to +1, indicating respectively from a strong negative correlation to a strong positive correlation.

2.1.1.1.2 Other weighting metrics

Apart from Pearson's correlation coefficient, other metrics, such as adjusted cosine similarity, Spearman's rank correlation coefficient, or mean squared difference measure are also proposed to calculate user-based similarity. However, empirical study made by Herlocker et al [35] shows that for user-based recommender system, the Pearson coefficient outperforms other measures.

Still, the "pure" Pearson measure alone is not ideal as there are cases that the measure cannot handle. Consider a real life problem that there are some items that are favored by everyone, Pearson's measure would not consider that an agreement by two users on a controversial item has more weight than an agreement on a universally like item. Herlocker et al [35] also showed that applying the measure to user who has rated very few items also lead to bad predictions. Therefore, many attempts, such as significance weighting proposed by Herlocker et al [35], or case amplification suggested by Breese et al [16] have been made to fill the gap and improve the accuracy. However, the question of whether these weighting schemes are helpful in real-world settings is still opened.

2.1.1.1.3 Challenges

Although user-based approaches have been deployed successfully, they faces serious challenges when are applied to large e-commerce sites which possess millions of users and items. Specifically, the cost for scanning a vast number of potential neighbors makes it impossible for the system to predict in real time. Therefore, large scale e-commerce sites often opt for other techniques, one of them is the item-based nearest neighbor approach.

2.1.1.2 Item-based nearest neighbor recommendation

The main idea of this approach is to calculate the similarity between items instead of one between users. The advantage of this approach over user-based approach is that we can preprocess an item similarity matrix that characterize the degree of similarity between items. At run time, a prediction for product p and user u is made by detecting the most relevant items using the item similar matrix and by calculating the weighted sum of u's rating for these items. As the number of relevant items is commonly limited, the computation of the prediction can be done within a short time frame, suitable for such online applications. A similar matrix is, theoretically, also possible with user-based approaches; however, in real time scenarios, the number of overlapping ratings for two random users is relatively small, making it unstable as a few more ratings may significantly affect the similarity between users.

For item-based approaches, the cosine similarity is found to be the standard metric [39]. The cosine similarity is defined as follows:

$$sim(\overrightarrow{a}, \overrightarrow{b}) = \frac{\overrightarrow{a} \cdot \overrightarrow{b}}{|\overrightarrow{a}| * |\overrightarrow{b}|}$$

where $\overrightarrow{a}$ is an item vector, $\cdot$ denotes the dot product, and $|\overrightarrow{a}|$ is the Euclidian length of the vector, which is defined as the square root of the dot product of the vector with itself.

One drawback of cosine measure is that it does not take into account the fact that different users have different rating schemes, i.e., some users rate items highly in general, while some others give lower ratings. This drawback is solved using adjusted cosine measure, which subtracts the user average from the rating. Let U be the set of users that rate both item a and b. The adjusted cosine measure is as follows:

$$sim(a, b) = \frac{\sum_{u \in U}(r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,a} - \bar{r}_u)^2}\sqrt{\sum u \in U(r_{u,b} - \bar{r}_u)^2}}$$

with $\bar{r}_u$ is the average rating for item u.

The prediction function, which is computed in real time, is defined as follows:

$$pred(u, p) = \frac{\sum_{i \in ratedItem(u)} sim(i, p) * r_{u,i}}{\sum_{i \in ratedItem(a)} sim(i, p)}$$

Compare to user-based recommendation, item-based approaches prove to be often more scalable as for most of the case, the number of items falls behind the number of user, thus it requires less time and space to compute the similarity matrix (if necessary). Also, item-based approaches are more justifiable by users, for they can easily grasp the explanation of the recommendation, modify the list of neighbors and alter the weights. User-based methods, on the other hand, are less able to justify, as recommendations come from other users are hard to explain. However,

as item-based approaches are based on ratings on similar items, the recommender tend to suggest items that might be already familiar to that user. While this behavior leads to safe recommendations, it does not help users explore other novel items that they might like as well.

In general, nearest neighbor approaches work well for popular items. Nonetheless, there are two important drawbacks with these approaches when deal with unpopular item:

- Limited coverage: both approaches define neighbor as having ratings in common. This assumption is limiting, as users with very few common items can still have similar preferences. Moreover, coverage of such approaches can be limited, as only items rated by neighbor can be recommended.
- Sensitive to sparse data: For most system, users only rated a small portions of available items. This results in a cold star problem, when some items have very few or no ratings at all, which affects the prediction of these items. Another problem is when there are only few ratings, the weight of each item has a significant impact over the similarity between vectors, reducing the accuracy of the recommender.

To solve these problem, many small tunes for the neighborhood approaches have been made, such as the use of Significance Weighting [] for the weighting problem or ... for the Cold Start problem []. Besides that, other advance algorithms have also been developed to tackle these topics. One of the most popular approach that has gained attraction recently is the use of matrix factorization, as it was exploited to significantly improve the accuracy of the recommender system in the Netflix Prize competition in 2009.

2.1.1.3 Matrix factorization/ latent factor model

Matrix factorization methods is used to derive a set of salient patterns from user-ratings. For example, let "Gone with the wind" and "Me before you" be the set of liked item of user A, and "Romeo and Juliet" and "The fault in our star" be the set of liked item of user B, while nearest neighbor approaches would consider these books separately, matrix factorization could see that all these books belong to romantic genre and therefore recommend them to the other user. The factors, however, is not always obvious. In some cases, they can be uninterpretable.

The idea of this method is to factorize the original sparse matrix into a product of matrices. Each decomposed matrix is much denser than the original one. The technique can be used for both similarity matrix and rating matrix. There are many matrix factorization techniques with increasing complexity and accuracy. For the scope of this thesis, I will describe the Singular value decomposition (SVD) model, a basic yet effective decomposition technique. The example is an adaptation from the one from Grigorik [85].

Consider the table 2.1:

SVD takes a m-by-n matrix M and decomposes it into three factors: two unitary matrices U and V, which represent the user matrix and the item matrix accordingly, and a nonnegative diagonal matrix Σ. The main point of this decomposition is that after receiving the product matrix, we can retain only the most important values in the diagonal matrix to build back the approximation of the original matrix. The decomposition is as follows:

	User 1	User 2	User 3	User 4
Item 1	3	4	3	1
Item 2	1	3	2	6
Item 3	2	4	1	5
Item 4	3	3	5	2

TABLE 2.1: Rating for SVD-based recommendation

$$M = U\Sigma V^T$$

where V^T is the transpose matrix of V.

Applying the decomposition, we obtain $\Sigma = 12.2215, 4.9282, 2.0638, 0.2977$ and the two matrices

U			
-0.4312	0.4932	-0.5508	-0.5172
-0.5327	-0.5305	0.4197	-0.5085
-0.5237	-0.4052	-0.4873	0.5693
-0.5059	0.5578	0.5321	0.3871

V			
-0.3593	0.3677	-0.2961	0.8050
-0.5675	0.0880	-0.6285	-0.5246
-0.4429	0.5686	0.6590	-0.2150
-0.5939	-0.7306	0.2882	0.1746

In this case, we can eliminate the two dimensions with the lowest values and only keep the two dimensions with value $\Sigma = 12.2215, 4.9282$. Figure 2.1 is the projection of the two matrices in a two-dimensional space.

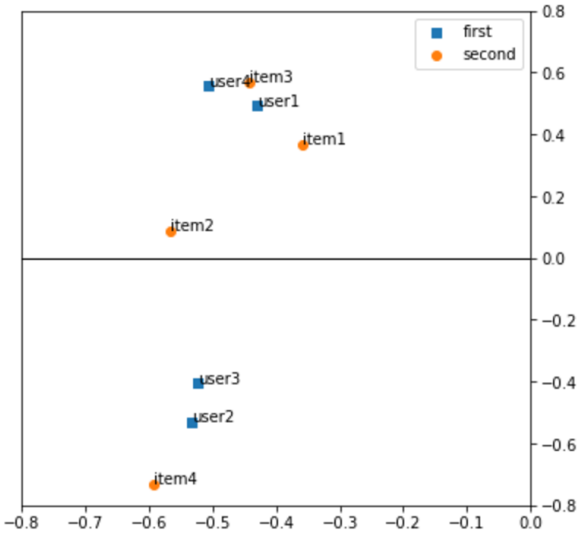

FIGURE 2.1: Projection of user and item on a two-dimensional space

After the product matrices are constructed, a new user profile can be added by multiply user's rating with the user matrix U and the diagonal matrix Σ. Once the user profile is constructed, many strategies can be used for the recommendation of item to that user. One possible approach is to again apply cosine similarity measure

to find neighbor user. Another approach is to approximate user rating by exploiting the interaction between user and item in the latent factor space [43].

2.1.2 Content-based recommendation

Although collaborative filtering approaches work well, it still have some limitations. Apart from the cold star problem and the popularity bias, collaborative approaches require lots of user rating for the system to be stable. Besides, these approach cannot take advantage of the semantic content of the item. For example, it would be obvious to recommend "Gone with the wind" to a user, if we know that (a) this book belongs to romantic catalog and (b) the user has an affair for romantic novel. Therefore, as data is becoming more abundant, new approaches have been studied to exploit these data for the recommendation process. These approaches are commonly called content-based recommendation.

The basic idea of content-based approaches is to classify similar items using a list of features of each item. For example, a book recommender could use the metadata of the book, such as the author's name and the book's genre as a reference for similarity; or it could calculate the similarity of two books based on the overlap of keywords, using the Dice coefficient [72] as follows:

$$sim(b_i, b_j) = \frac{2 * |keywords(b_i) \bigcap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

with $|keywords(b_i)|$ is the number of keyword in book b_i.

However, metadata also need the clarification of expert. The standard approach in content-based recommendation is, not to maintain a list of metadata, but to seek for the similarity in the content of the item itself. In the case of book recommender, one common approach is to transform the content of a book into a vector in a multi-dimensional Euclidian space using *term frequency-inverse document frequency (TF-IDF)* technique. For the scope of this thesis, the detail of the technique is not described here.

Music recommender systems, however, take a different approach compare to other text recommender sytems, as the differences in characteristics between the two items. Metadata of a music track can include general tags generated by expert such as genre, artist, album, etc..., as well as other social tags by users; and the content of a track is the acoustic features that are analyzed from the signal of that track, of which the most representative ones are timbre and rhythm [19].

Basically, a content-based recommender has three basic components [71]

- Content analyzer: the mission of this component is to extract relevant information from the content of the item. In this phase, feature extraction techniques are used to exploit information and transform them to the representation that the recommender needs. The representation is the input to the *profile learner* and *filtering component* parts.
- Profile learner: in this phase, the module tries to construct user profiles using the representation of the item. Often, this phase is achieved through the use of various kind of machine learning techniques [58], depending on the nature of input data. User profiles are then passed into filtering component for generating list of recommendations.
- Filtering component: This phase generates a list of recommendation for the users by comparing the similarity between the representation of user profiles

and that of items to be recommended. After recommendation is suggested, the system might get feedback of the users for the profile learner phase to improve the user profiles.

A survey of the techniques used in content-based recommender will be discussed later. For the moment, some music features as well as the techniques that are used to extract them will be described. The profile learner and filtering component parts will not be detailed here, as they are beyond the scope of the thesis. However, the specific algorithms used for these part will be described later in chapter 3.

2.1.2.1 Feature extraction

Music, in its original form, is a record of analog audio signal (i.e, electrical voltage) [37]. To store music digitally, a analog-to-digital conversion is needed. The process has two phases: sampling and quantization [41]. In the sampling part, a signal is sampled by evaluating its amplitude at a particular time, with the number of samples taken per second is called the sampling rate. The amplitude measurements are then mapped as 8 or 16-bit integers, whose process is called quantization.

Because digital music is represented as a series of integers, the raw information they contain is trivial for human at the perceptual level. Therefore, the first step of a music recommender is to extract useful features from the raw representation. In music domain, different taxonomies have been created to capture audio features in certain perspectives. Weihs et al. [93] divided audio features into four subcategories, including short term features, long term features, semantic features, and compositional features. Another taxonomy was proposed by Scaringella [73], which separates audio into three different components: timbral to denote features related to spectral content (i.e. shape) of the signal; temporal features such as loudness, tempo, onset rate; and tonal component such as harmonic, pitch, key, scale, and chords distribution.

Fu et al. [29] developed another taxonomy under human understanding of music perspective. According to the hierarchy, there are two level of audio features: low-level and mid-level features. Low-level features are features that can be obtained directly from the audio with proper signal processing techniques like Fourier transform, spectral analysis, autoregressive modeling, etc. Timbral and temporal are two main classes in low-level features. Low-level features have been exploited massively in music classification, due to the simple procedure to obtain and their good performance. However, they are not closely related to the nature of music that human perceive. Mid-level features, on the other hand, delimitate music using rhythm, pitch, and harmony, concepts that are more familiar to normal listeners.

2.1.2.1.1 Low-level features

Timbral is perhaps the most exploited characteristic in the set of low-level features. It is described as the quality of sound, with different timbres belong to different types of sound sources, e.g. different instruments. Table 2.2 lists some major timbre features. Despite of the large variety in number of features, the timbre extraction phase of all these features are closely related to each other and follow some standard procedures. As this thesis does not deal with low level signal processing, the detail of these procedures is not described here.

Timbral features have been successfully used in the past for genre classification. However, it still has several disadvantages. One critical problem is that with the technological advance in the recording process (e.g., tape editing, equalization, and

Class	Feature Type	Used in
Timbre	Zero Crossing Rate	[89], [47], [12]
	Spectral Centroid	[89], [47], [12], [59]
	Mel-frequency Cepstrum Coefficient	[89], [12], [53]
	Fourier Cepstrum Coefficient	[12], [50]
	Stereo Panning Spectrum Features	[90], [91]
Temporal	Statistical Moments	[89], [47]
	Amplitude Modulation	[64], [62]
	Auto-Regressive Modeling	[82], [57]

TABLE 2.2: Low-level audio features

compression), the final signals of the post production stage of the same instrument in two different tracks would be different. As timbral features techniques rely heavily on raw signal, they are heavily affected by the recording process, of which phenomenal called album effect [96].

Temporal features are another low-level features that are used to capture the temporal evolution of the signal. The difference between the two features is that, while each timbral is extracted in a local window of raw signal with 10- 100 ms duration, temporal extraction is performed on a series of timbre features in larger frames, allowing for the derivation of a richer set of features, such as fluctuation pattern [64], rhythmic pattern [48], rhythmic coefficient [94], etc. Morchen [59] has generated a set of operation that can be employed on top of timbre to produce new features at a coarser scale. Many of these features belong to the amplitude modulation family, as they are generated by analyzing the modulation of the amplitude spectrum.

2.1.2.1.2 Mid-level features

Low-level features were dominantly applied for genre classification; however, they do not capture the intrinsic properties of music that humans perceive. Mid-level features, such as rhythm, pitch, or harmony, are more familiar to human. Therefore, they play an important role in some specific domains, such as query by singing [38] or detect cover versions of popular songs with similar melodies [88]

Rhythm is the most widely used mid-level feature in audio-based music classification. It describes the recurrence of tension and release in music. From rhythm, "danceability" of a track can be derived by . Rhythm can also be used for mood classification [27] [97], since sad songs usually have a slow rhythm, while exciting songs usually possess a fast rhythm.

Pitch is another important mid-level feature. It is defined as the most fundamental frequency of the sound. Often, a pitch histogram is constructed and is combined with low-level feature for genre and mood classification [89] [46]. Pitch can also used to develop pitch class profile and harmonic pitch class profile, which in turn are useful for detecting similar melodies and transcription [54] [67].

Harmony is a succession of musical chords, which are three or more notes, typically sounded simultaneously. The chord is detected by compare pitch histogram with chord template to identify the possible chords. Chord features are often used as a complimentary to pitch features in detecting melody similarity and cover song [26] [10]

To summarize, low-level audio features such as timbre are sufficient for genre classification but fail to achieve good result in song similarity detection. Mid-level features, on the other hand, are successful in detecting similar song using pitch and

harmonic features. Rhythm features have also been exploited for mood classification. Research in using acoustic contents for recommendation problems will be describe later in the overview of music recommender systems part.

2.1.3 Hybrid recommendation

Hybrid recommendation is a method that combine several kind of recommenders together. The motivation for hybrid recommender is to try to take advantage of the strength of different algorithms with fewer drawbacks than any individual one. Burke's well-known taxinomy [17] differentiate between seven kind of hybridization strategies; However, the seven approaches can be abstracted into three base design: monolithic, parallelized, and pipelined hybrids [39]. Monolithic design incorporates several recommendation strategies into one algorithm, while parallelized and pipelined designs require at least two separate recommenders. The outlines of the three designs are depicted in figure 2.2, 2.3 and 2.4

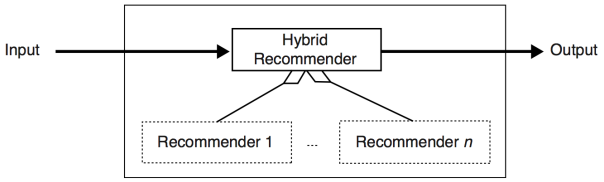

FIGURE 2.2: Monolithic hybridization design

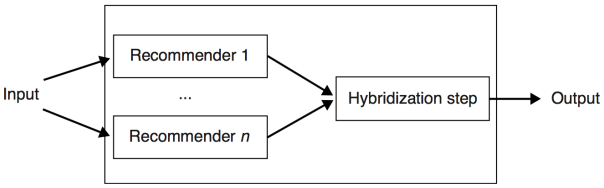

FIGURE 2.3: Parallelized hybridization design

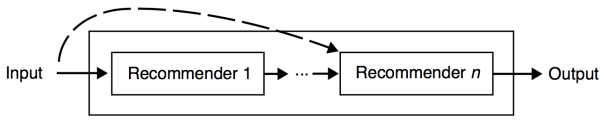

FIGURE 2.4: Pipelined hybridization design

2.1.3.1 Monolithic hybridization design

Monolithic design consist of a single recommender that combines different type of input data and and uses a build-in algorithm to exploit these data for the recommendation process. Feature combination and feature augmentation strategies from Burke's taxonomy [17] can be assigned to this category.

A feature combination hybrid is a monolithic recommender that simply combine information from a varied source of input data. For instance, Basu et al. [9] proposed a hybrid recommender that combines collaborative features, such as users' likes and

dislikes of each movie, with genre features of the movie. Therefore, instead of identifying just a group of users who watch the same movies, the recommender determines a group of users who watch same movies of the same genre, such as comedy, drama, or action.

A feature augmentation hybrid is also a monolithic design that combines different source of features. In contrast with feature combination, feature augmentation does not simply combine features, but also apply some transformations so that some features are strengthened as compared to other features. Content-boost collaborative filtering [56] is an example of this hybrid. The algorithm creates a concept of pseudo-user-ratings, as well as uses different weighting scheme between collaborative features and content features.

2.1.3.2 Parallelized hybridization design

Parallelized hybridization designs gather result of several recommenders and aggregate their outputs using a hybridization machanism. Mixed, weighted, and switching are three main strategies that are used in the parallel design.

A mixed hybrid design combines the results of several recommenders at the level of user interface. The top-scoring items are then presented to users in order. However, as items from different recommenders are mixed together, some conflicts might happen. For example, Zanker et al. [99] built a mixed system to propose bundles of different product in the tourism domain. One bundle, for example, contains two recommenders, one for accommodations and the other for sport and leisure activities. As the two recommenders are separated, there would be a case when the top-score accommodations are far away from the suggested sport place. Therefore, a Constraint Satisfaction Problem solver is often employed in the design to resolve these kind of conflicts.

A weighted hybrid design also combines the results of different recommenders, then compute the weighted sums of their scores. The weight can be estimate by using an empirical bootstrapping approach [100] or a dynamic weighting scheme [23].

Switching hybrid strategy assigns which recommender to use under certain situations. For example, to overcome the cold start problem, the system can use the results of content-base recommender until enough user rating are available. When the recommender see a sufficient number of user rating, it starts to use results generated from collaborative filtering recommender. One practical example of this hybrid design is the NewsDude system [13] to recommend news articles. First, a content-based nearest neighbor recommender is used. If there is no related articles found, a user collaborative filtering recommender is used to propose cross-genre alternative.

2.1.3.3 Pipelined hybridization design

Pipelined hybrids is a design that the output of a recommender is used as the input of another one with the exception of the final one, whose results are shown to users. Cascade hybrids and meta-level hybrids are two types of this principles.

In cascade hybrid design, the succeeding recommender can only receive the output of its predecessor. In other words, except the first recommender that process the whole set of items, all other remaining recommenders can only alter the score or exclude the items that endure from their preceders without the ability to add new one. As each recommender have the property of reducing the size of the recommending set, cascade strategies might meet a problem of not proposing enough recommending items. The problem can be solved by combining cascade hybrids with

a switching strategy, as applied by Zanker et al. [100] to build a cigar recommender system.

A meta-level hybrid is one that, in the training phase, use a recommender to transform raw data to a model that is then fed into the actual recommender. In the testing phase, however, the contributing recommender is eliminated from the system. That is to say, the raw data in the testing phase is evaluated directly by the actual recommender. This hybrid design is hard to apply, as the contributing recommender has to generate a model that is useful for the actual recommender, and not all recommendation logics can do so.

2.1.4 Other kind of recommendation

So far, collaborative filtering, content-based filtering, and hybrid approaches have been introduced. There are many other kinds of recommendation system, such as knowledge-based recommendation, critique-based recommendation that are used in some specific domains. In the music domain, there are also context-aware recommender system, where the recommenders try to incorporate contextual information, such as time and current activities, into the recommendation process. I will not discuss in deep these methods here, as it goes beyond the scope of the thesis. However, information of recommenders that use contextual information will also be included in the overview of music recommender systems part.

In the next section, recommender systems that are build in music domain will be discussed. Specifically, the kind of information as well as the methods used in content-based and context-based recommendation will be presented. Also, some hybrid designs that combine content, context, and user-rating information will also be introduced.

2.2 Overview of music recommender systems

Music, differs from other content domains such as books or movies, has its own unique characteristics regarding consumption. For example, the consumption time of books and movies are quite lengthy, ranging on the average of few hours (for movies) to several days (for books); songs, on the other hand, takes a much shorter time for listener to consume, only a few minutes. Consequently, this lead to the ephemeral and disposable nature of music. Another example is the number of repetition: a single song can be consumed repeatedly (even multiple times in a row), while books are movies are consumed a few times at most. This implies that the user might appreciate recommendations of items that they already heard in the past.

In the past, many approaches, based on the observation of the nature of music and listener's behaviors, had been tried to build an efficient music recommender system. Till now, there are three main approaches to such system [71], which will be described in the following sections.

2.2.1 Content-Based Music Recommendation

Content-based recommendation exploits information describing music as material for recommender systems. There are two main approaches for content-based system: one uses metadata, such as annotations or social tags, while the other analyses audio content using machine learning techniques.

2.2.1.1 Metadata content

Metadata comes in several forms. One is the manual annotations constructed by music experts or voluntary community. This kind of annotation often obeys a strict structured taxonomy build by expects. Another form of annotations is social tags, which is built by asking casual users to provide unstructured text annotations for the item. The last kind of annotations is information collected on web pages, blogs and RSS feeds related to music items.

2.2.1.1.1 Annotation

Manual annotation contain information such as musical genre, record label, year, knowledge about tracks and artists, and albums. Some musical properties, particularly tempo, mood, and instrumentation can also be added. Many online database have been built using editorial metadata, following by many recommender systems trying to exploit them.

Bogdanov et al. [14] build an artist recommender using exclusively metadata from *Discogs*, a free and community-built database containing information about artists, records labels, and their releases. For each artist in the database, a tag weight vector is created using genre, style, label, country, and year information of the releases related to the artist. The role of the artist in each release (e.g. main artist, track artist, or extra artist) and the relations between artist, such as aliases and membership relations, are also taken into account. A sparse tag matrix is then formed from the artist vectors, and latent semantic analysis [25] is applied to reduce the dimension of the matrix. Afterwards, the authors use Pearson correlation distance [30] to measure the similarity between artists.

Apart from community-built database, some other database are developed by experts for commercial use. *Pandora*, for example, is a personalized radio that built its recommender using annotations done by experts [40]. *AllMusic* is another example that also provides mood annotations besides general editorial metadata. However, not much research has been done using these database, as they are proprietary, and there is no public data sets of this kind are available for researcher. Constructing such a data set would be costly, and they are difficult to scale to large collections.

2.2.1.1.2 Social tags

Social tags are tags provided by user using the services. Tags are personalized, arbitrary, and do not follow any particular structure, ranging from genre like "blue" or "jazz" to event-related attributes (e.g. "live") and assertion (e.g. "my favorite song"). They also vary in scope, from broad one such as "classical" to niche terminology like "Malcolm Arnold" or "renaissance". Social tags, therefore, need to be preprocess for the data to be useful. A popular method, which is used by *Last.fm*, a social music website, for structuring social tags is to transform them into a folksonomy [83]. The tag weight vectors technique is then applied to compute the similarity [32], with the enhancement by using latent semantic analysis techniques to overcome vector sparsity problem [45]

2.2.1.1.3 Annotations by web crawling

Apart from tags made by experts and social tags, some recommender systems are built using information crawled from web pages. These recommenders often apply artist similarity metric, generated by using text mining techniques [78], as the main

principle for the recommendation. Green et al. [32] compute artist-to-artist similarity using keyword extracted from *Wikipedia* entries and social tags from *Last.fm*. Similarly, McFee and Lanckriet [55] predict artist similarity based on social tags and keyword extracted from artist biographies on *Last.fm*. A deviant approach is the one made by Lim et al., as they compute song-level similarity through bag-of-words representations of lyrics found on *musiXmatch.com* [49]

2.2.1.2 Audio content

Audio content analysis is promoted by MIR researchers as an alternative to metadata and collaborative filtering method [8]. Content analysis is expected to solve "long tail" problem, where unpopular music items are not suggested because the lack of available user ratings, tags, and other types of metadata [22]. Music content is separated into two broad categories: acoustic features taken directly from the audio, and semantic annotations derived from acoustic features using machine learning techniques.

2.2.1.2.1 Acoustic features

Acoustic features are properties of a sound that can be recorded and analyzed using signal processing techniques. As mentioned in the content-based recommendation part, there are three main features that are often used by recommender system: timbral features, temporal and time-domain features, and tonal features.

Timbral similarity method converts timbre information to a standard representation and applies a number of methods to approximate the likelihood between two songs [4] [52]. For example, Logan [51] builds a recommender that compares Mel-frequency cepstrum coefficient (MFCC) based distance of user's music set with target play list. The approach, however, is insufficient as evaluation shows a nominal number of customer satisfaction [15].

Pampalk et al. [62] [63] study an algorithm that use, in addition to spectral similarity, loudness fluctuations and two derived descriptors from sound wave to improve the accuracy of music similarity and genre classification of a song. The algorithm is used to recommend user playlists, in which songs that are similar to the ones that are skipped by user are eliminate, and only tracks that are similar to the ones that the user wholly listens to remains.

Celma and Herrera [21] take another approach, calculating Euclidean distance using timbre, dynamics, tempo, meter, tonal strength, key, and mode information. This method is compared to an item-based collaborative filtering and a hybrid method on a large scale evaluation. The result shows that all three algorithms work fine recommending familiar items. For unfamiliar items, collaborative filtering reveals a poor discovery ratio, while pure content-based method sometimes goes off direction, as the recommender confuse between similar sounds, such as between the sound of classical guitar and the one of harpsichord). The hybrid method performs the best, as it limit the number of possible tracks whose artists are related with the original artist, therefore reducing mistakes performing by pure content-based method.

2.2.1.2.2 Semantic annotation

Pure acoustic signal, unfortunately, cannot directly capture semantic meaning of a track. As a result, mere acoustic feature recommenders do a poor job in song suggestion, as an "energetic" song can be recommended next to a "nostalgic" track because

of the similarity of the instrument. A sudden change in genre might frustrate customer, as one would not expect a mourning song in a middle of a party. Therefore, many approaches have been made in order to bridge this semantic gap by using machine learning techniques to predict annotations from audio content.

However, extracting genre or mood information from acoustic content is perplexing, as mapping between human annotation and acoustic cannot be clearly defined [5]. To solve this, Barrington et al. [8] propose a method to measure semantic similarity: they train Gaussian mixture models of MFCCs for semantic concepts such as genres, moods and instruments. Therefore, for a song, a distribution of tags is generated, which is then compared to another in order to estimate similarity.

2.2.2 Contextual Music Recommendation

The vast majority of existing recommender system approaches focus on information about users and items but not context information, such as time and place of the event. Only until recently, the topic of context-awareness starts to gain attraction in recommender system research [2]. Context, in the fields that are directly related to recommender system, is defined as "information describing where you are, whom you are with, and what resources are nearby" [80]. Therefore, context in the domain of music can be derived as a collection of factors that affects user's appreciation of music, such as time, mood, and current activities.

Contextual information can be classified using various kind of classifications. Adomavicius et al. [2] suggest categorizing context into three distinct classes: fully observable, partially observable, and unobservable context. Dey and Abowd [1], in attempt to construct another classification, propose to classify context into primary context, which is four most important factors that describe user situation: location, identity, activity, and time; and secondary context, which is data derived from primary information. Applying the classification of Dey and Abowd, M. Schedl et al [79] divide context information into two generic classes: environment-related context and user-related context. Environment-related context refers to information that can be obtained by user's computer or mobile phone, such as location, time, weather, etc., while user-related context indicates information that can be derived from the environment-related one, such as user's activity, emotion state, and social environment.

2.2.2.1 Environment-Related Context

Surrounding environment has been proven to have an influence on user's preference of music. Adrian C. North and David J. Hargreaves [60] find a correlation between musical descriptors, such as arousal, sensuality, spirituality, and listening situation, such as activity, spirituality, and social constraint. Pettijohn et al. [66] find that different seasons, such as winter and summer, also affect musical preferences.

Many attempts have been made to build recommender using environment-related information. Reddy and Mascia [69] used space information capturing using GPS coordinates, internal time data, kinetic information derived from difference in GPS signals, and even meteorological info to build a recommender for a mobile music player called Lifetrak. Songs have to be tagged manually by users using system predefined tagging system, and are played in appropriate situation based on users' preferences. For example, the app may play rock music when a user is in a gym, and classical music when the user tries to study in a cafeteria.

Ankolekar and Sandholm from HP labs [3] propose a mobile audio application, Foxtrot, that exploits crowd-sourced geo-tagged audio information to provide

a stream of location-aware audio content to the users. The recommender, however, generated poor user experiences, as an environment generates different meaning to different people, leading to diverse music preferences. Indeed, a research made by Okada et al. [61] shows that not only the algorithm, but also the architectural design and usability of the application that matter, as user feedback suggests the need for explanations of the recommendations and more control over the playlist.

2.2.2.2 User-Related Context

One's music preference is not only affected by geological and activity component, but also by factors such as emotions and social background. Schäfer and Sedlmeier [75] discovered that one's music preference is also linked with one's sociocultural and physiological functions. In other words, people use music preference as a mean to express their identities and personal values. User-related context can be divided into the following groups:

- Activity information: information implies user's actions (e.g., walking, driving, working) or user's state (e.g., walking pace or heart rate). Foley [28] showed that people with different occupation have different favored music tempo. For example, those who work with power machines like a slow allegro, while typists prefer faster tempo like presto.
- Emotional information: current mood of user has a direct impact on the choice of music. For example, a user may want to listening energetic music while he is happy, and calm music vice versa. Schäfer and Sedlmeier [75] found that music has a function to moderate listener's mood by energizing him or making he feel better.
- Social context information: music preference can be affected by the presence of other people. People may choose music taken into account the event they participate in. Many researchers have address the issue of group recommender system. For example, Popescu and Pu [68] proposed using probabilistic weighted sum as the algorithm to recommend group playlist.
- Cultural context information: information about user's culture characteristics. Koenigstein et al. [42] exploited file sharing information on a Peer to Peer network in US to predict the success of a song on Billboard Hot 100 Chart. Schedl [77] built a location-aware recommender system by retrieving geo-tagged Twitter tweets to detect listening trend at a particular place.

Compare to environment-related context, user-related context is more difficult to infer using electronic devices. Many attempts have been made to predict user's emotion or daily activities [65] [92] by extracting environment-related feature such as the time of day, temperature, weather, etc... Emotion-based music recommender has gained attention recently, due to advances in automatic music emotion recognition [98]

2.2.3 Hybrid music recommendation

Since music preference is a complex and multi-faceted concept, it is reasonable to incorporate multiple aspects of music similarity into recommendation. Like other hybrid designs applied in other domains, music hybrid designs are also expected to

gain better performance than a single approach. There are three main hybrid system in music domain: one that combines content and context information, one that combines collaborative filtering and content, and the final one mixes collaborative filtering and context information.

2.2.3.1 Hybrid designs combining content and context information

To date, not so many research focus on combining these two types of information. One typical example is the Mobile Music Genius project [76], which gather context information, such as time, location, weather, motion, and player-related features, and metadata, such as genre and artist, and use a decision tree to adapt the playlist as the user's context changes by a certain amount.

Another mobile music recommender built by Wang et al. [92] also uses a hybrid design. The music player collects information about time of the day, accelerometer data, and ambient noise to predict user's current activity, such as running, walking, sleeping, etc. The player then associate activity information with activity labels of music tracks to infer suitable playlist for the activity.

2.2.3.2 Hybrid designs combining collaborative filtering and content information

Hybrid designs combining these two features receive massive intention from researchers, as the two features fusing together is expected to solve multiple problems [17] [18] [20]

- Cold start problem: As audio content analysis can be performed instantly to new track, content-based recommender can be used to recommend songs for a user when his profile is not sufficient enough for collaborative filtering techniques.
- Popularity bias: collaborative filtering might cause popularity bias, as users only listen to popular items. Including tracks with similar content to the recommendation is a solution for this problem.
- novelty and diversity: popularity bias results in a limited range of recommendation. Content-based recommendation helps suggest more tracks from the long tail, therefore increasing the novelty and diversity of the system.

Techniques applied in these hybrid designs can be a simple one, such as ensemble learning or fusing output, to a more sophisticated one, such as the one that use probabilistic framework or graph-based interpretation. The content features used in these research varied from only low-level features (MFCC), to a combination of low and mid-level features (timbral, temporal, and tonal features), and also with track metadata (genre, mood, release year, etc.)

One simple example is a hybrid design that uses ensemble learning [86]. The system implements a item-item collaborative filtering recommender as well as a content-based recommender using timbral, tempo, genre, mood, and release year information. Both recommenders use the weighted mean of most similar item's rating to predict the scores of an item. The feature combination strategy is then applied to associate the two scores into a vector for each user, and the mapping $\{P_{r_1}, \ldots, P_{r_n}\} \rightarrow R$ is constructed, with R_{r_n} is the estimate of the nth recommender and R is the actually observed value. To predict a value for a new instance, first, a feature vector $\{P_{r_1}, \ldots, P_{r_n}\}$ for that instance is established. Then the Euclidean

distance is calculated between this vector and all the predicted vectors to find the most similar vector, whose actual value is promoted.

2.2.3.3 Hybrid designs combining collaborative filtering and context information

The third kind of hybrid design tries to combine collaborative and context features. One example is the context-aware music recommender built by Baltrunas et al. [7]. The recommender try to offer context-aware music to users in a car scenario by extending matrix factorization model with contextual parameters.

Another example a context-aware music recommendation based on latent topic sequential patterns by Hariri et al. [33]. The recommender use latent factor techniques to discover latent topics, and sequential pattern mining techniques to reveal the relationship between these topics. The prediction then predicts the next topic using a sequence of song in user's current playlist.

Chapter 3

Implementation

In this chapter, the approaches used in this research is examined. First, the join dataset merging from Last.fm and Spotify is mentioned. After that, the three algorithms and the evaluation metrics are discussed in detail.

3.1 Dataset

The dataset for this thesis is a combination between two databases from two services: the 1K users dataset from Last.fm [1] and the dataset crawling from Spotify Web Api [2].

The dataset from Last.fm contains the listening habit from February, 2005 until May, 2009 for nearly 1000 users. The dataset contains more than 19 million entries, each entry consists of 4 attributes: the user id from Last.fm, the song that the user listened to, the artist who wrote the song, and the timestamp represents the time the user listened to the song.

From the last.fm dataset, top 10000 songs that have the highest listening frequency are choosen. Then, track name and the respective artist are queried on Spotify API to get the corresponding track features. As Spotify API not only returns exact match but also similar match, and the name convention between the two systems might be different, a fuzzy matching algorithm is applied to filter the results. Particularly, the Levenshtein ratio is applied on the two fields. Any matching with a ratio higher than 80 percent is a good match, since the two strings are basically the same with some minor differences, which are the results of the different convention between the two system. Since matches with ratio between 60 and 80 percent might be good ones, manual check was done to guarantee the best result. I decided to cut off the result at 60 percent, as the majority of matches that are less than 60 percent are noisy and not accurate.

As some of the songs from Last.fm have yet been analyzed by Spotify, after joining the two datasets, the number of remaining songs is about 4200. I then remove users with less than 10 listening events. The dataset is then divided into training set and test set following this rule: for each user, split the listening time into two halves, of which the first half belongs to the training set and the second half belongs to the test set. In the test set, all the tracks that are listened only 1 time are eliminated, as it is likely that the user only listened to it by chance. The final dataset contains with 891 users, 4200 unique tracks, and ...
The final dataset contains the following fields:

- Username: The user who listen to the album, taken from last.fm database.

[1] https://last.fm
[2] https://developer.spotify.com/web-api

- Album: The name of the album the user listens to.
- Artist: The artist who performs the album.
- Playcount: The total number of time the user listens to the album.
- TrackId: The unique id of the track in Last.fm dataset, which also serves as unique key.
- Energy: A perceptual measurement of intensity and activity in a range from 0 to 1. The higher the score, the higher the energy the track contains. For example, metal rock has high energy, while a Bach prelude is perceived to have low energy. This measurement is built on top of dynamic range, loudness, timbral, onset rate, and general entropy.
- Speechiness: A measurement of how much spoken words is present compare to music. A value from 0.66 to 1 indicates that the track is mostly spoken words, such as talk show or audio book. Vice versa, a value between 0.33 and 0.66 implies that the track contains both music and speech, ranging from pop to rap music. A value below 0.33 indicates a non speech track.
- Acousticness: A evaluation of how much acoustic a track contains compare to how much electronic. Again, a value closer to 1 implements that the track is played with mostly acoustic instruments, such as guitar and harmonica; while a value closer to 0 implements the present of electronic instruments.
- Danceability: A measurement of how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. The higher the value, the more danceable the track.
- Tempo: The overall estimated tempo of a track, measuring in beats per minute (BPM).
- Instrumentalness: The assessment of how much a track incorporates vocals. The higher the instrumentalness, the greater likelihood the track contains no vocal content.
- Key: The key (tonal center) a track is in. Integers map to pitches using Pitch Class notation [3].
- Valence: A measure from 0 to 1 of the positiveness of a track. Tracks with high valence sound more positive (e.g. happy, cheerful), and vice versa.
- Liveness: An indication of whether a track is performed live or in studio. The higher the liveness, the higher possibility that the track is performed live.
- Loudness: The overall loudness of a track in decibels (dB), with range from -60 to 0 db.
- Mode: An indication of modality (major or minor) of a track. The field has only 2 values: 1 for major and 0 for minor.
- Time signature: An indication of how many beats are in each bar of a track.

[3] https://en.wikipedia.org/wiki/Pitch_class

3.2 Algorithms

The following sections describe my implementation of the three algorithms that are used in this thesis: a Naïve collaborative filtering, a collaborative filtering approach specified for implicit feedback dataset, followed by the hybrid approach that combine collaborative filtering with item features.

3.2.1 Pure Collaborative Filtering

As the number of song outnumbers the number of user, I decide to implement the user-user based collaborative filtering. The algorithm can be summarized in the following steps:

1. Calculate similarity between users using cosine similarity between users' rating vectors.
2. For each user, form a neighborhood of *n* users that have the highest similarity.
3. Compute a prediction for an item from a weighted combination of selected neighbor's ratings.

In step 3, predictions are calculated using the weighted average of deviations from neighbors' mean. The formula is as follow:

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \overline{r}_u) \times P_{a,u}}{\sum_{u=1}^{n} P_{a,u}} \tag{1}$$

with $p_{a,i}$ is the prediction of item i for user a, $P_{a,i}$ is the similarity between user a and user u, and $\overline{r}_a$ is the average of the ratings of user a.

3.2.2 Collaborative Filtering for Implicit Feedback Dataset

3.2.2.1 Introduction

One of the most well known work on implicit feedback dataset is the one of Yifan Hu et al [**hu2008collaborative**]. In his paper, he mentioned that almost all the prior approaches were mostly designed for explicit feedback, and therefore applying these methods on implicit dataset might not work well. According to his work, there are four crucial differences between the two type of dataset:

1. No negative feedback: In explicit feedback dataset, the users actively tell us which items they like and the ones that they do not. In implicit feedback, however, telling the items that the users dislike is hard to achieve. The reason is that, by observing the user behaviors, we can infer that the user might like the items that they consume. However, for an item that the user has no interaction with, it is unknown whether the user does not favor it or he does not know about its existence. This phenomenon derives a crucial implication: in explicit dataset, a pair of user - item that has no information is treated as missing data and is omitted from the analysis. This is impossible for implicit dataset, as eliminating these missing data would leave us with only positive feedback, greatly misrepresenting the full user profile.
2. Implicit feedback is noisy: passively tracking user behaviors can only give us indirect information about user preferences and true motives. For example,

we may view purchase from a user, but this does not necessarily indicate a positive view of the product. The user might buy the item as a gift, or show disappointment only later the purchase. This leads to the third distinction.

3. The numerical value of explicit feedback indicates preference, while the numerical value of implicit feedback indicates confidence: Explicit feedback let users express their level of preference, e.g. a scale between 1 (totally dislike the item) to 5 (totally love the item). In contrast, numerical values of implicit feedback describe the frequency of actions, e.g. how much time the user listens to a particular song or watches a show. A large value not necessarily implies that the user like the item. For example, in the film domain, a user might really like a movie but is hesitated to watch it again because of the lengthy duration. However, he might periodically watch a weekend show, which derives from his habit but not appreciation of the show.

4. Evaluation of implicit-feedback recommender requires appropriate measures. Traditionally, metrics such as mean squared error are often applied to measure the success of the recommender. However, in implicit models we have to also consider the available of the items, or the competition between items. For example, it is unclear how to evaluate two shows that are on at the same time, and hence cannot be both watched by the user.

The above comments are used to make adjustments to traditional collaborative filtering for it to work with implicit dataset. The two main important of the new model, hence, is as follows:

1. The model transfers raw observation (r_{ui}) into two separate magnitudes with distinct interpretations: preference (p_{ui}) and confidence level (c_{ui}).

2. All possible user-item combinations $n\dot{m}$ have to be addressed.

3.2.2.2 Model Formalization

The formalization of the model is as follows:

Let r_{ui} be the observation value between user u and item i. In out situation, r_{ui} denotes the number of time user u listened to track i. For example, $r_{ui} = 2$ indicates that in the past, user u listens to track i 2 times. Next, we introduce a set of binary variable p_{ui}, which indicates the preference of user u and item i. The values of p_{ui} is defined by the formula:

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

We believe that if a user u listened to a track i, we have an indication that u likes i and vice versa. However, our belief is associated with different confident level. If the user has never listened to the track, the confident level is low as we cannot infer anything from the user. If the track is heard once or twice, we are also not so confident in claiming that the track is favored by the user, as the listening might be from autoplaying a whole album or a random playlist. However, as the number of listening time increase, we can also increase the confidence in saying that the user intentionally listen to the song, and therefore like it. Consequently, we introduce a set of variable c_{ui}, which measures the confidence in observing p_{ui}. A plausible choice for c_{ui} would be:

$$c_{ui} = 1 + \alpha r_{ui}$$

with α is the rate of increase.

We also define user factor matrix X and item factor matrix Y for matrix factorization: we fix a number k and represent each user u with a k dimensional vector x_u, and represent each item i with a k dimensional vector y_i. The $n \times k$ user matrix X is formed by the set of user $x_1, \cdots, x_n \in \mathbb{R}^k$), and the $m \times k$ item matrix Y is formed by the set of item $y_i, \cdots, y_m \in \mathbb{R}^k$:

$$X = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix} \quad Y = \begin{bmatrix} | & & | \\ y_1 & \cdots & y_m \\ | & & | \end{bmatrix}$$

Having introduce the variables, now our main goal is to find a user vector $x_u \in \mathbb{R}^k$ for each user u, and an item vector $y_i \in \mathbb{R}^f$ for item i so that the inner product between the two vector is approximately equal to the preference: $p_{ui} = x_u^T y_i$ with x^T is the tranpose of vector x. In other words, we would like to minimize the following cost function:

$$min_{x,y} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right) \tag{1}$$

with $\lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$ is the regularization to make sure that the model would not overfit the training data.

Often, for explicit feedback, Stocastic Gradient Descent (SGD) is used to minimize the cost function. However, as in implicit feedback we have to consider all possible u, i pairs, SGD cannot be apply. Therefore, another optimization process has been applied to minimize the cost function as follow.

Observing that when we fix either the user-factors or the item-factors, the cost function becomes quadratic and the global minimum can easily be reach. Therefore, we take the differentiation to find the analytic expression for x_u that minimizes the cost function (1):

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

with C^u is a diagonal $n \times n$ matrix where $C^u_{ii} = c_{ui}$ and $p(u)$ is the preference vector of user u. The naïve calculation will take $O(k^2 n)$ for each of the m user because of the computation $Y^T C^u Y$. The calculation can be speed up by first parse $Y^T C^u Y = Y_T Y + Y^T (C^u - I) Y$. As $Y_T Y$ is now independent of u, it can be computed beforehand. As for $Y_T (C^u - I) Y$, we can see that $C^u - I$ has only n_u non-zero elements where n_u is the number of item that $r_u i > 0$) and often $n_u << n$. Similarly, $C^u p(u)$ also contains just n_u non-zero element. Hence, the running time for $Y^T (C^u - I) Y$ is $O(k^2 n_u)$, the run time for inverting the matrix $(Y^T C^u Y + \lambda I)^{-1}$ is $O(k^3)$, and the total run time is $O(k^2 n_u + k^3)$ for each user. For m user, the total running time is $O(k^2 N + k^3 m)$, where $N = \sum_u n_u$.

Similar to user-factors, we can do the computation on all item-factors using the same technique. The analytic form for y_i is:

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i)$$

The running time for recomputing all item-factors is $O(k^2N + f^3n)$.

The pseudo code for the algorithm is in Algorithm 1:

Algorithm 1 ALS CF for implicit dataset

function SGD($T, \lambda, \{(x_y, y_t)\}_{i=1}^{m}$) // number of steps, regularization parameter, training set
  draw random indices $\pi_1, \ldots, \pi_T$
  $w \leftarrow 0$
  $b \leftarrow 0$
  for $t = 1, \ldots, T$ **do**
    $l \leftarrow \ell'(w \cdot x_{\pi_t} + b, y_{\pi_1})$
    $w\_initial \leftarrow 1 - (1/t) \cdot w$
    $b\_initial \leftarrow 1 - (1/t) \cdot b$
    $w\_deviate \leftarrow l \cdot x_{\pi_t}/\lambda t$
    $b\_deviate \leftarrow l/\lambda t$
    $w \leftarrow w\_initial - w\_deviate$
    $b \leftarrow b\_initial - b\_deviate$
  return [w, b]

3.2.3 Metadata Embeddings for Item Recommendation

In this section, I propose using a hybrid ranking matrix factorization model using k-Order Statistic Loss (k-OLS) to solve the recommendation problem. k-OLS is an improvement of the Weighted Approximate-Rank Pairwise (WARP) Loss, a ranking error function used to optimize precision at k.

This approach is chosen because of three advantages: first, it has the ability to exploit the side information, i.e. user features or item features, if there is any. Second, the loss function is design to optimize the ranking of the top k item. This advantage has a practical implication: consider Youtube website where there are not so many placeholder for recommended item, a high precision for top 5 - 10 items would play a critical role in user experience. The third advantage is that the loss function also treats missing feedback as unknown but not negative, which is different from the well known similar loss function WARP [95]. More detail about this will be explained in the algorithm description.

3.2.3.1 Metadata Embedding Hybrid Model

In this section, I will introduce the hybrid content-collaborative model [44]. In this model, like in a collaborative model, users and items are represented as latent vectors. These latent vectors are defined by functions of embedding of the content features that describe each product or user. For example, if the movie "Game of Throne" is described by the following features: "fantasy", "drama", and "adventure", then its latent representation is the sum of these features' latent representations.

The structure of the model is motivated by two considerations:

- The model must have the ability to learn user and item representations from the interaction data: if two items are constantly liked by users, the model must know that the two items are alike.
- The model must be able to compute recommendations for new users and items

The first motivation is filled by using the embedding feature factors. If two items are constantly liked by users, their embedding feature factors would be similar. Otherwise, if two items are never liked by the same user, their embedding factor would be different from each other. This feature has an implication: if a user u likes one of the two items, we can confidently recommend the other.
The second motivation is achieved by representing user and item as linear combination of the content features. Therefore, if a new movie with the features "fantasy", "drama", and "adventure" is added to the system, the representation vector of the movie is the sum of the vectors of the three features.
The formalization of the model is as follow: Let U be the set of m users, I be the set of$n$ items, F^U be the set of user features, and F^I be the set of item features. Each user u is described by a set of features $f_u \subset F^U$, and each item i is described by a set of features $f_i \subset F^I$. The two user feature and item feature matrices are then parameterized into $m \times d$ and $n \times d$ embedding matrices $\boldsymbol{e}^U$ and $\boldsymbol{e}^I$ respectively as follows:

$$e_f^U = \begin{bmatrix} — & \boldsymbol{e}_1^U & — \\ & \vdots & \\ — & \boldsymbol{e}_m^U & — \end{bmatrix} \quad e_f^I = \begin{bmatrix} — & \boldsymbol{e}_1^I & — \\ & \vdots & \\ — & \boldsymbol{e}_n^I & — \end{bmatrix}$$

Similarly, we define two bias matrices b^U and b^I. The latent representation of user u is the sum of its features' latent vectors:

$$\boldsymbol{q}_u = \sum_{j \in f_u} \boldsymbol{e}_j^U$$

The same for item i:

$$\boldsymbol{p}_i = \sum_{j \in f_i} \boldsymbol{e}_j^I$$

The bias term for user u is the sum of its bias latent vectors:

$$b_u = \sum_{j \in f_u} b_j^U$$

The same applies for item bias term i:

$$b_i = \sum_{j \in f_i} b_j^I$$

The prediction for user u and item$i$ is the dot product between two corresponding latent representation plus the two bias terms:

$$f_i u = f(\boldsymbol{q}_u \cdot \boldsymbol{p}_i + b_u + b_i) \tag{2}$$

There are many suitable functions for $f(\cdot)$. As mentioned in the beginning of the section, I will apply a ranking approach, thus use the prediction function to try to infer the rank of an arbitrary item.

3.2.3.2 Learning to Rank Recommendation with WARP Loss

Given a set of items for a user, a ranking function is responsible for returning an ordered list of items, with the most relevant items are arrange at the top of the list, therefore have a lower rank. In order to do so, the system is given a training with a list of users as well as the set of their known ratings. The ratings can be explicit, i.e. user ratings, or implicit, i.e. number of times a user listened to a track, or watched a movie. In our case, we use listening time as the rating for the ranking function, and the Weighted Approximate-Rank Pairwise (WARP) loss [95] as our chosen ranking error functions. The main idea of this method is that, we first define the tracks that have been listened by a user positive items for that user, and the remaining negative items. Then for each positive item, we examine a number of random negative items. For each negative item, if the rank of the item is lower than the one of the positive item, we perform a gradient update to lower the loss function.

Formally, let D_u be the set of the positive items for user u, we define $f(u)$ to be the vector of all item scores $1, \ldots, |D|$ for the user u. Therefore, the item score for an item d is defined as follow:

$$f_d(u) = \frac{1}{|D_u|} \sum_{i \in D_u} (\boldsymbol{q}_u \cdot \boldsymbol{p}_i + b_u + b_i)$$

where $\boldsymbol{q}, \boldsymbol{p}, b_u, b_i$ are mentioned in equation (2).

To learn f, we minimize an object function of the following form:

$$\sum_{u=1}^{|U|} L\left(f(u), D_u\right) \tag{3}$$

where L is the loss function between the known ratings D_u and the predictions $f(u)$ for user u.

The class of Weighted Approximate-Rank Pairwise (WARP) loss functions is then defined as follow:

$$L_{WARP}(f(u), D_u) = \sum_{d \in D_u} \Phi\left(rank_d(f(u))\right) \tag{4}$$

where $rank_d(f(u))$ is the rank of the true label d given by $f(u)$:

$$rank_d(f(u)) = \sum_{\bar{d} \notin D_u} I\left(f_{\bar{d}}(u) \geq f_d(urank_d\left(f\left(u\right)\right))\right)$$

and $\Phi(\cdot)$ transforms this rank into a loss:

$$\Phi(k) = \sum_{j=1}^{k} a_j, \text{ with } a_1 \geq a_2 \geq \ldots \geq 0$$

This class of functions allows one to define different choices of $\Phi(\cdot)$ for different minimize purpose. Minimizing Φ with $a_j = \frac{1}{Y}$, with Y is the total number of item would optimize the mean rank; while setting $a_1 = 1$ and $a_{j>1} = 0$ would optimize only the first rank. Generally, setting high value of a for the first k item would optimize the top k item in the ranked list, leading to optimize precision at k.

To minimize the loss function (4), the authors derive a differentiable approximation of the rank and use stochastic gradient descent (SGD) to minimize the approximation instead of directly minimize the rank itself, as the rank contains integer values and therefore indifferentiable. To begin with, the loss function of a positive document d is equal to:

$$L_{WARP}(f_d(u), D_{d_u}) = \sum_{\bar{d} \notin D_u} \Phi\left(rank_d\left(f\left(u\right)\right)\right) \frac{I\left(f_{\bar{d}}\left(u\right) \geq f_d\left(u\right)\right)}{rank_d\left(f\left(u\right)\right)}$$

with the convention $0/0 = 0$ when the correct label d is top-ranked. As ranks are integer number, hinge loss is applied to replace the indicator function to add a margin and make the loss continuous. The loss function is then approximated by:

$$\overline{L_{WARP}}($$

3.2.3.3 Stochastic Gradient Descent

SGD is an optimization technique to minimize the linear predictor $(\mathbf{w}, b)$. Given the randomize initial predictor $(\mathbf{w_0}, b_0)$, the algorithm will perform T gradient descend steps and produce a sequence of intermediate linear predictors $((\mathbf{w}_t, b_t))_{t=0}^{T}$. In each gradient descend step t, the linear predictor is updated using the information exploiting from an individual training example (x, y).
Formally, let $\pi_1, \ldots, \pi_T$ be a sequence of independently random indices, with $1 \leq \pi_t \leq m$. On each iteration t, the algorithm runs on index π_t. Also, let $[\mathbf{w}, b]$ denote the concatenation of $\mathbf{w}$ and b. The subgradient of Eq. (2) is :

$$\nabla \mathbf{F}(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \ell'(\mathbf{w} \cdot x_i + b, y_i)[x_i, 1] + \lambda[\mathbf{w}, b] \tag{3}$$

As π is a random index, chosen uniformly between 1 and m,

$$\lambda[\mathbf{w}, b] + \ell'(\mathbf{w} \cdot x_\pi + b, y_\pi)[x_\pi, 1]$$

is an unbiased estimator of Eq. (3), as well as a stochastic gradient of the objective function in Eq. (2). We choose the size of learning step t to be $1/\lambda t$, where $lambda$ is the regularization parameter in Eq. (1). The learning step size is chosen according to the theoretical convergence analysis of SGD with strongly convex objective functions [34]. Therefore, the update of the predictor on each iteration t is:

$$[\mathbf{w}_t, b_t] = [\mathbf{w}_{t-1}, b_{t-1}] - \frac{1}{\lambda t}\left(\ell'(\mathbf{w}_{t-1} \cdot x_{\pi_t} + b_{t-1}, y_{\pi_t})[x_{\pi_t}, 1] + \lambda[\mathbf{w}_{t-1}, b_{t-1}]\right)$$

Rearrange the above equation:

$$[\mathbf{w}_t, b_t] = \left(1 - \frac{1}{t}\right)[\mathbf{w}_{t-1}, b_{t-1}] - \frac{\ell'(\mathbf{w}_{t-1} \cdot x_{\pi_t} + b_{t-1}, y_{\pi_t})}{\lambda t}[x_{\pi_t}, 1] \tag{4}$$

The pseudo code for this algorithm is in Algorithm 1:

3.2.4 Content-boosted Collaborative Filtering

The idea of the content-booasted collaborative filtering algorithm is quite simple. First, a pseudo user-ratings vector is created for every user in the database. Each

Algorithm 2 SGD for regularized linear regression

```
1: function SGD(T, λ, {(x_y, y_t)}_{i=1}^m) // number of steps, regularization parameter, training set
2:     draw random indices π_1, ..., π_T
3:     w ← 0
4:     b ← 0
5:     for t = 1, ..., T do
6:         l ← ℓ'(w · x_{π_t} + b, y_{π_1})
7:         w_initial ← 1 − (1/t) · w
8:         b_initial ← 1 − (1/t) · b
9:         w_deviate ← l · x_{π_t}/λt
10:        b_deviate ← l/λt
11:        w ← w_initial − w_deviate
12:        b ← b_initial − b_deviate
       return [w, b]
```

element of a pseudo user-ratings vector of a user is the user-rating if available, or the value predicted by the content-based recommender otherwise. The formal definition of a pseudo vector is as follow:

$$v_{u,i} = \begin{cases} r_{u,i} & : \text{ if user } u \text{ rated item } i \\ c_{u,i} & : \text{ otherwise} \end{cases}$$

when $r_{(u,i)}$ denotes the actual rating of user u for item i, and $c_{(u,i)}$ is the prediction of the content-based predictor.

The pseudo user-ratings vectors of all user together construct a dense pseudo user-rating matrix. This matrix is then used as a substitution of the original user-rating matrix for collaborative filtering method.

To increase the accuracy of the model as well as to promote the score from the content predictor, the following weighting schemes are added:

Harmonic Mean Weighting: The accuracy of a pseudo user-ratings vector depends on the number of album the user listens to. The more album and the more times the user listen to them, the better the result that the content-based predictor generates and vice versa. Therefore, *Harmonic Mean weighting* is used to reduce the impact of pseudo vector that does not contain much actual user information. The weighting is as follow:

$$hm_{i,j} = \frac{2m_i m_j}{m_i + m_j}$$

$$m_i = \begin{cases} \frac{n_i}{T} & : \text{ if } n_i < T \\ 1 & : \text{ otherwise} \end{cases}$$

with n_i is the number of item user i has rated. The harmonic mean tends to bias toward the lower value; therefore, if one of the two vectors has less than T items, the correlation will be devalued accordingly.

ld threshold T and e experience to de- ve T

Significant Weighting: If a user has a high correlated neighbor but the number of co-rating between the two users are small, it is a high chance that the neighbor would

worsen the prediction for the user. Therefore, a *Significant weighting* is imposed to the correlation between two users. The weighting penetrates the correlation by a factor of n/T if the number of co-rated item is less than T; otherwise, the correlation is left unchanged. The equation for the weighting is as follow:

$$sg_{a,u} = \begin{cases} \frac{n}{T} & : \text{ if } |r_a \cap r_u| < T \\ 1 & : \text{ otherwise} \end{cases}$$

with r_a is the set of rating item of user a. The final hybrid correlation weight $hw_{a,u}$ is as follow:

$$hw_{a,u} = hm_{a,u} + sg_{a,u} \tag{5}$$

Self Weighting: The idea of self weighting is that we want to put more weight in the content-based prediction as we have more user listening information and vice versa. The formula for self weighting is as follow:

$$sw_a = \begin{cases} \frac{n_a}{T} \times max & : \text{if} n_a < T \\ max & : \text{otherwise} \end{cases} \tag{6}$$

where n_a is the number of items rated by the user. The parameter max is the maximum confidence that we put on the content-based predictor, in case the user listen to more than T album. In our experiments, we used a value of 2 for max

Final Predictions Combining the above weighting schemes, the final hybrid prediction for a user a for item i is as follow:

$$p_{a,i} = \overline{v}_a + \frac{sw_a \left(c_{a,i} - \overline{v}_a\right) + \sum_{\substack{u=1 \\ u \neq a}}^{n} hw_{a,u} P_{a,u} \left(v_{u,i} - \overline{v}_u\right)}{sw_a + \sum_{\substack{u=1 \\ u \neq a}}^{n} hw_{a,u} P_{a,u}}$$

where $c_{a,i}$ is the content-based prediction for user a for item i; $v_{u,i}$ is the pseudo user-rating for user u and item i; $\overline{v}_u$ is the mean over all item for user u. sw_a, $hw_{a,u}$, and $P_{a,u}$ are shown in equation (6), (5), and (1) respectively; and n is the size of neighborhood.

Bibliography

[1] Gregory Abowd et al. “Towards a better understanding of context and context-awareness”. In: *Handheld and ubiquitous computing*. Springer. 1999, pp. 304–307.

[2] Gediminas Adomavicius and Alexander Tuzhilin. “Context-aware recommender systems”. In: *Recommender systems handbook*. Springer, 2011, pp. 217–253.

[3] Anupriya Ankolekar and Thomas Sandholm. “Foxtrot: a soundtrack for where you are”. In: *Proceedings of Interacting with Sound Workshop: Exploring Context-Aware, Local and Social Audio Applications*. ACM. 2011, pp. 26–31.

[4] J-J Aucouturier, François Pachet, and Mark Sandler. “" The way it Sounds": timbre models for analysis and retrieval of music signals”. In: *IEEE Transactions on Multimedia* 7.6 (2005), pp. 1028–1035.

[5] Jean-Julien Aucouturier. “Sounds like teen spirit: Computational insights into the grounding of everyday musical terms”. In: *Language, evolution and the brain* (2009), pp. 35–64.

[6] Marko Balabanović and Yoav Shohom. “Content-based, callaborative recommendation”. In: *Communications of the ACM* 40.3 (1997).

[7] Linas Baltrunas et al. “Incarmusic: Context-aware music recommendations in a car”. In: *E-Commerce and web technologies* (2011), pp. 89–100.

[8] Luke Barrington, Reid Oda, and Gert RG Lanckriet. “Smarter than Genius? Human Evaluation of Music Recommender Systems.” In: *ISMIR*. Vol. 9. 2009, pp. 357–362.

[9] Chumki Basu, Haym Hirsh, William Cohen, et al. “Recommendation as classification: Using social and content-based information in recommendation”. In: *Aaai/iaai*. 1998, pp. 714–720.

[10] Juan Pablo Bello. “Audio-Based Cover Song Retrieval Using Approximate Chord Sequences: Testing Shifts, Gaps, Swaps and Beats.” In: *ISMIR*. Vol. 7. 2007, pp. 239–244.

[11] Jacob Benesty et al. “Pearson correlation coefficient”. In: *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[12] James Bergstra et al. “Aggregate features and AdaBoost for music classification”. In: *Machine learning* 65.2-3 (2006), pp. 473–484.

[13] Daniel Billsus and Michael J Pazzani. “User modeling for adaptive news access”. In: *User modeling and user-adapted interaction* 10.2-3 (2000), pp. 147–180.

[14] Dmitry Bogdanov and Perfecto Herrera. “Taking advantage of editorial metadata to recommend music”. In: *9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*. 2012, pp. 618–632.

[15] Dmitry Bogdanov et al. “Semantic audio content-based music recommendation and visualization based on user preference examples”. In: *Information Processing & Management* 49.1 (2013), pp. 13–33.

[16] John S Breese, David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering". In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1998, pp. 43–52.

[17] Robin Burke. "Hybrid recommender systems: Survey and experiments". In: *User modeling and user-adapted interaction* 12.4 (2002), pp. 331–370.

[18] Robin Burke. "The adaptive web. chapter Hybrid web recommender systems". In: (2007).

[19] Pedro Cano, Markus Koppenberger, and Nicolas Wack. "An industrial-strength content-based music recommendation system". In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2005, pp. 673–673.

[20] áOscar Celma. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer, 2010.

[21] Òscar Celma and Perfecto Herrera. "A new approach to evaluating novel recommendations". In: *Proceedings of the 2008 ACM conference on Recommender systems*. ACM. 2008, pp. 179–186.

[22] Òscar Celma Herrada et al. *Music recommendation and discovery in the long tail*. Universitat Pompeu Fabra, 2009.

[23] Mark Claypool et al. "Combing content-based and collaborative filters in an online newspaper". In: (1999).

[24] Dan Cosley et al. "Is seeing believing?: how recommender system interfaces affect users' opinions". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2003, pp. 585–592.

[25] Scott Deerwester et al. "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6 (1990), p. 391.

[26] Daniel PW Ellis and Graham E Poliner. "Identifyingcover songs' with chroma features and dynamic programming beat tracking". In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 4. IEEE. 2007, pp. IV–1429.

[27] Yazhong Feng, Yueting Zhuang, and Yunhe Pan. "Music information retrieval by detecting mood via computational media aesthetics". In: *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*. IEEE. 2003, pp. 235–241.

[28] John P Foley Jr. "The Occupational Conditioning of Prefer Ential Auditory Tempo: A Contribution toward an Empirical Theory of Aesthetics". In: *The Journal of Social Psychology* 12.1 (1940), pp. 121–129.

[29] Zhouyu Fu et al. "A survey of audio-based music classification and annotation". In: *IEEE transactions on multimedia* 13.2 (2011), pp. 303–319.

[30] Jean Dickinson Gibbons and Subhabrata Chakraborti. "Nonparametric statistical inference". In: *International encyclopedia of statistical science*. Springer, 2011, pp. 977–979.

[31] David Goldberg et al. "Using collaborative filtering to weave an information tapestry". In: *Communications of the ACM* 35.12 (1992), pp. 61–70.

[32] Stephen J Green et al. "Generating transparent, steerable recommendations from textual descriptions of items". In: *Proceedings of the third ACM conference on Recommender systems*. ACM. 2009, pp. 281–284.

[33] Negar Hariri, Bamshad Mobasher, and Robin Burke. "Context-aware music recommendation based on latenttopic sequential patterns". In: *Proceedings of the sixth ACM conference on Recommender systems*. ACM. 2012, pp. 131–138.

[34] E. Hazan, A. Agarwal, and S. Kale. "Logarithmic regret algorithms for online convex optimization". In: *Machine Learning* 69.2-3 (2007), pp. 169–192.

[35] Jonathan L Herlocker et al. "An algorithmic framework for performing collaborative filtering". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1999, pp. 230–237.

[36] Will Hill et al. "Recommending and evaluating choices in a virtual community of use". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co. 1995, pp. 194–201.

[37] Jay Hodgson. *Understanding Records: A Field Guide to Recording Practice*. Bloomsbury Publishing, 2010.

[38] Jyh-Shing Roger Jang and Hong-Ru Lee. "A general framework of progressive filtering and its application to query by singing/humming". In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.2 (2008), pp. 350–358.

[39] Dietmar Jannach et al. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[40] Nicolas Jones and Pearl Pu. "User technology adoption issues in recommender systems". In: *Proceedings of the 2007 Networking and Electronic Commerce Research Conference*. HCI-CONF-2008-001. 2007, pp. 379–394.

[41] Daniel Jurafsky and James H Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*.

[42] Noam Koenigstein, Yuval Shavitt, and Noa Zilberman. "Predicting billboard success using data-mining in p2p networks". In: *Multimedia, 2009. ISM'09. 11th IEEE International Symposium on*. IEEE. 2009, pp. 465–470.

[43] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems". In: *Computer* 42.8 (2009).

[44] Maciej Kula. "Metadata embeddings for user and item cold-start recommendations". In: *arXiv preprint arXiv:1507.08439* (2015).

[45] Mark Levy and Mark Sandler. "Learning latent semantic models for music from social tags". In: *Journal of New Music Research* 37.2 (2008), pp. 137–150.

[46] Tao Li and Mitsunori Ogihara. "Detecting emotion in music". In: (2003).

[47] Tao Li, Mitsunori Ogihara, and Qi Li. "A comparative study on content-based music genre classification". In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM. 2003, pp. 282–289.

[48] Thomas Lidy et al. "Improving Genre Classification by Combination of Audio and Symbolic Descriptors Using a Transcription Systems." In: *ISMIR*. 2007, pp. 61–66.

[49] Daryl Lim, Brian McFee, and Gert R Lanckriet. "Robust structural metric learning". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 615–623.

[50] Chien-Chang Lin et al. "Audio classification and categorization based on wavelets and support vector machine". In: *IEEE Transactions on Speech and Audio Processing* 13.5 (2005), pp. 644–651.

[51] Beth Logan. "Music Recommendation from Song Sets." In: *ISMIR*. 2004, pp. 425–428.

[52] Beth Logan and Ariel Salomon. "A Music Similarity Function Based on Signal Analysis." In: *ICME*. 2001, pp. 22–25.

[53] M Mandel and D Ellis. "Song-level features and SVM for music classification". In: *In Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR*. Vol. 5. 2006.

[54] Matija Marolt. "A Mid-level Melody-based Representation for Calculating Audio Similarity." In: *ISMIR*. 2006, pp. 280–285.

[55] Brian McFee and Gert Lanckriet. "Learning multi-modal similarity". In: *Journal of machine learning research* 12.Feb (2011), pp. 491–523.

[56] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations". In: *Aaai/iaai*. 2002, pp. 187–192.

[57] Anders Meng et al. "Temporal feature integration for music genre classification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.5 (2007), pp. 1654–1664.

[58] Tom M Mitchell. "Machine learning. 1997". In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

[59] F Morchen et al. "Modeling timbre distance with temporal statistics from polyphonic music". In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (2006), pp. 81–90.

[60] Adrian C North and David J Hargreaves. "Situational influences on reported musical preference." In: *Psychomusicology: A Journal of Research in Music Cognition* 15.1-2 (1996), p. 30.

[61] Karla Okada et al. "ContextPlayer: Learning contextual music preferences for situational recommendations". In: *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications*. ACM. 2013, p. 6.

[62] Elias Pampalk, Arthur Flexer, Gerhard Widmer, et al. "Improvements of Audio-Based Music Similarity and Genre Classificaton." In: *ISMIR*. Vol. 5. London, UK. 2005, pp. 634–637.

[63] Elias Pampalk, Tim Pohle, and Gerhard Widmer. "Dynamic Playlist Generation Based on Skipping Behavior." In: *ISMIR*. Vol. 5. 2005, pp. 634–637.

[64] Elias Pampalk, Andreas Rauber, and Dieter Merkl. "Content-based organization and visualization of music archives". In: *Proceedings of the tenth ACM international conference on Multimedia*. ACM. 2002, pp. 570–579.

[65] Han-Saem Park, Ji-Oh Yoo, and Sung-Bae Cho. "A context-aware music recommendation system using fuzzy bayesian networks with utility theory". In: *International Conference on Fuzzy Systems and Knowledge Discovery*. Springer. 2006, pp. 970–979.

[66] Terry F Pettijohn, Greg M Williams, and Tiffany C Carter. "Music for the seasons: seasonal music preferences in college students". In: *Current psychology* 29.4 (2010), pp. 328–345.

[67] Graham E Poliner et al. "Melody transcription from music audio: Approaches and evaluation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.4 (2007), pp. 1247–1256.

[68] George Popescu and Pearl Pu. "Probabilistic game theoretic algorithms for group recommender systems". In: *Colocated with ACM RecSys 2011 Chicago, IL, USA October 23, 2011* (2011), p. 30.

[69] Sasank Reddy and Jeff Mascia. "Lifetrak: music in tune with your life". In: *Proceedings of the 1st ACM international workshop on Human-centered multimedia*. ACM. 2006, pp. 25–34.

[70] Paul Resnick. "An Open Architecture for Collaboative Filtering of Netnews". In: *Proc. of CSCW'94*. 1994, pp. 175–186.

[71] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook". In: *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[72] Gerard Salton. "Syntactic approaches to automatic book indexing". In: *Proceedings of the 26th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 1988, pp. 204–210.

[73] Nicolas Scaringella, Giorgio Zoia, and Daniel Mlynek. "Automatic genre classification of music content: a survey". In: *IEEE Signal Processing Magazine* 23.2 (2006), pp. 133–141.

[74] J Ben Schafer, Joseph A Konstan, and John T Riedl. "Recommender Systems for the Web". In: *Visualizing the Semantic Web, Springer* (2006), pp. 102–123.

[75] Thomas Schäfer and Peter Sedlmeier. "From the functions of music to music preference". In: *Psychology of Music* 37.3 (2009), pp. 279–300.

[76] Markus Schedl. "Ameliorating music recommendation: Integrating music content, music context, and user context for improved music retrieval and recommendation". In: *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*. ACM. 2013, p. 3.

[77] Markus Schedl. "Leveraging Microblogs for Spatiotemporal Music Information Retrieval." In: *ECIR*. Springer. 2013, pp. 796–799.

[78] Markus Schedl et al. "Exploring the music similarity space on the web". In: *ACM Transactions on Information Systems (TOIS)* 29.3 (2011), p. 14.

[79] Markus Schedl et al. "Music recommender systems". In: *Recommender Systems Handbook*. Springer, 2015, pp. 453–492.

[80] Bill Schilit, Norman Adams, and Roy Want. "Context-aware computing applications". In: *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*. IEEE. 1994, pp. 85–90.

[81] Upendra Shardanand and Pattie Maes. "Social information filtering: algorithms for automating "word of mouth"". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co. 1995, pp. 210–217.

[82] JS Shawe-Taylor and Anders Meng. "An investigation of feature models for music genre classification using the support vector classifier". In: (2005).

[83] Mohamed Sordo et al. "The quest for musical genres: Do the experts and the wisdom of crowds agree?" In: *ISMIR*. 2008, pp. 255–260.

[84] *Spotify Acquires The Echo Nest*. 2014. URL: https://press.spotify.com/us/2014/03/06/spotify-acquires-the-echo-nest/.

[85] *SVD recommendation system in ruby*. http://www.igvita.com/2007/01/15/svd-recommendation-system-in-ruby/. Accessed: 2017-08-15.

[86] Marco Tiemann and Steffen Pauws. "Towards ensemble learning for hybrid music recommendation". In: *Proceedings of the 2007 ACM conference on Recommender systems*. ACM. 2007, pp. 177–178.

[87] Shari Trewin. "Knowledge-based recommender systems". In: *Encyclopedia of library and information science* 69.Supplement 32 (2000), p. 180.

[88] Wei-Ho Tsai, Hung-Ming Yu, Hsin-Min Wang, et al. "Query-By-Example Technique for Retrieving Cover Versions of Popular Songs with Similar Melodies." In: *ISMIR*. Vol. 5. 2005, pp. 183–190.

[89] George Tzanetakis and Perry Cook. "Musical genre classification of audio signals". In: *IEEE Transactions on speech and audio processing* 10.5 (2002), pp. 293–302.

[90] George Tzanetakis, Randy Jones, and Kirk McNally. "Stereo Panning Features for Classifying Recording Production Style." In: *ISMIR*. 2007, pp. 441–444.

[91] George Tzanetakis et al. "Stereo panning information for music information retrieval tasks". In: *Journal of the Audio Engineering Society* 58.5 (2010), pp. 409–417.

[92] Xinxi Wang, David Rosenblum, and Ye Wang. "Context-aware mobile music recommendation for daily activities". In: *Proceedings of the 20th ACM international conference on Multimedia*. ACM. 2012, pp. 99–108.

[93] Claus Weihs et al. "Classification in music research". In: *Advances in Data Analysis and Classification* 1.3 (2007), pp. 255–291.

[94] Kristopher West et al. "Novel techniques for audio music classification and search". In: *ACM SIGMultimedia Records* 1.1 (2009), pp. 15–15.

[95] Jason Weston, Samy Bengio, and Nicolas Usunier. "Wsabie: Scaling up to large vocabulary image annotation". In: *IJCAI*. Vol. 11. 2011, pp. 2764–2770.

[96] B. Whitman, G. Flake, and S. Lawrence. "Artist detection in music with minnowmatch". In: *Neural Networks for Signal Processing XI, Proceedings of the IEEE Signal Processing Society Workshop*. IEEE. 2001, pp. 559–568.

[97] Dan Yang and Won-Sook Lee. "Disambiguating Music Emotion Using Software Agents." In: *ISMIR*. Vol. 4. 2004, pp. 218–223.

[98] Yi-Hsuan Yang and Homer H Chen. "Machine recognition of music emotion: A review". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.3 (2012), p. 40.

[99] Markus Zanker, Markus Aschinger, and Markus Jessenitschnig. "Development of a collaborative and constraint-based web configuration system for personalized bundling of products and services". In: *Web Information Systems Engineering–WISE 2007* (2007), pp. 273–284.

[100] Markus Zanker and Markus Jessenitschnig. "Case-studies on exploiting explicit customer requirements in recommender systems". In: *User Modeling and User-Adapted Interaction* 19.1 (2009), pp. 133–166.