A Dialog System for Music Recommendation

Jia Chen , **Leilan Zhang**
Department of Computer Science and Technology, Tsinghua University
cj18@mails.tsinghua.edu.cn
zll17@mails.tsinghua.edu.cn

Abstract

In this project, we have designed, implemented and analyzed a dialogue system for music recommendation. We analyzed thousands of songs collected online to find correlations between users and songs. The system is composed of two main parts: context-aware recommendation module and chat module. Users can interacte with the system through text-based dialogue, and the system will provide music recommendations according to the preference of user learned from the previous chat history. This paper will illustrate the motivation of this project, the problems we faced, the methods we proposed and the functions we have implemented, results and their analysis.

1 Introduction

The rapid development of internet and movile device makes it possible for us to access abundant music resources freely. On the other hand, the number of songs available exceeds the listening capacity of individual. People often find it difficult to choose songs that matches their taste from millions of songs. Therefore, these are strong needs of a good recommendation system which can help people to discover music by giving quality recommendation.

At present, there are many music streaming services, like Pandora, Spotify. Those systems usually record users' interactive actions, analyze their preferences based on collaborative filtering algorithm to provide recomendations. Those system using this method can give high-precision recomendations and achieve great success. However, we hope that users could interact with the system in a more natural way and get more sense of participation, they could chat with the system using natural language directly and would obtain what they want from the system. This explains the motivation to build this system.

The system is composed of three main parts: the music intent detection module (M_0), the music recommendation module (M_1'), and the chat module (M_4). When a user interacts with the system, the module M_0 will start to detect whether there exists a music intent in user's utterance or not. If the utterance is determined as music-related, the module M_1' will be invoked to provide recommendation services for the user. Otherwise, the system will assume that the user only wants to chat, then it will call the module M_4 to reply normal response and chat with the user. In practice, the system is packaged as a serive running on a host, users can enjoy this service through the WeChat interface.

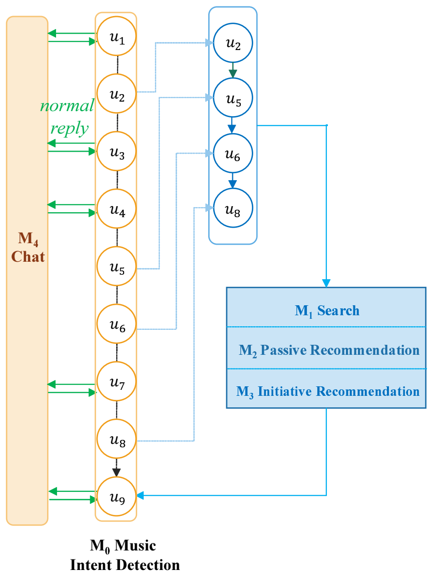
The rest of the paper is organized as follow. In Section 2, we describe related work, including the studies on music recommendation and graph representation learning. Section 3 describes in detail our approaches to build the system, including the data processing steps, the architecture of chat module M_4 and the submodules of the recommendation module M_1'. The experimental results for both chat module and recommendation effects are reported in Section 4. Finally, Section 5 presents our conclusions and future work plans.

2 Related Work

The two main approaches in music recomendation are collaborative filtering and content-based filtering [Song *et al.*, 2012]. Collaborative filtering technique has been proposed to recommend items through the choice of similar usres[Hill *et al.*, 1995]. Collaborative systems generate recomendation by comparing ratings of items between other users. These systems are built on the assumption that users who rate items similarly in the past will continue to rate them similarly in the future. The problem for collaborative systems is that they needs a huge amount of data about the users and the items to give recommendations. The systems would be ineffective if the data is not available or is too sparse. Moreover, in terms of the field of music, the assumption of collaborative filtering has not been widely studied yet, and problems like cold start are inevitable. On the other hand, content-based systems build models of the items and compare them to the preference model of the current user[Soleymani *et al.*, 2015]. The user's preference model records the user's response to each of the components in the item model. For instance, if the user has a positive response to songs of a specific style consistently, the preference model will likely indicate a positive bias towards this genre. The main challenge of these systems is that it depends on the correctness of the item models. This kind of system can not improve its understanding towards the music regardless of how much user data is collected.

3 Method

As mentioned in introduction setction, the system contains three main parts: the music intent detection module (M_0), the music recommendation module (M_1'), and the chat module (M_4). The music recomendation module M_1' also consists of three submodules: the search module M_1, the passive recommendation module M_2, and the initiative recomendation module M_3. Figure 1 demonstrates the whole architecture of the system.



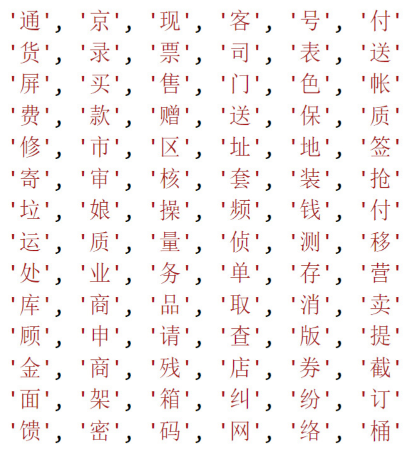
M_4 Chat
normal reply
u_1
u_2
u_3
u_4
u_5
u_6
u_7
u_8
u_9
M_1 Search
M_2 Passive Recommendation
M_3 Initiative Recommendation
M_0 Music Intent Detection

Figure 1: Architecture of the whole system

When the user send an utterance without any music intent, M_0 will call the chat module M_4 to response user's utterance. Otherwise, if the user query with clear and explicit music intent (for example, attempting to know the singer or the lyrics of a song), then the M_0 module will be employed to process the search task; if the user's utterance can reflect a certain degree of musical intent (for instance, looking for Kurt Cobain's Rock style songs), then M_0 will call the M_2 module to deal with the passive recommendation job; once the user give an utterance with inexplicit music intent, like "Any song for me?", then the M_3 module will be invoked to do the initiative recommendation work, it will recommend playlists according to user's chat history and feedbacks.

3.1 Data Process

For chat module, we collected dialogue data from both Jingdong dialogue corpus and Dilogue Action corpus. The Jingdong dialogue data consists of 129402 query-response pairs collected from conversations between the customer service staffs and the customers on Jingdong platform. This corpus provide high-quality data for dialogue generation. However, there are a lot of terms related to electronic shopping in this corpus (eg. "out of stock","consult","withdraw" etc.), so we mannually summarized those patterns and filtering out those domain-specific words. Figure 2 displays those moved words.



```
'通', '京', '现', '客', '号', '付'
'货', '录', '票', '司', '表', '送'
'屏', '买', '售', '门', '色', '帐'
'费', '款', '赠', '送', '保', '质'
'修', '市', '区', '址', '地', '签'
'寄', '审', '核', '套', '装', '抢'
'垃', '娘', '操', '频', '钱', '付'
'运', '质', '量', '侦', '测', '移'
'处', '业', '务', '单', '存', '营'
'库', '商', '品', '取', '消', '卖'
'顾', '申', '请', '查', '版', '提'
'金', '商', '残', '店', '券', '截'
'面', '架', '箱', '纠', '纷', '订'
'馈', '密', '码', '网', '络', '桶'
```

Figure 2: Filter words for JD dialogue Data

The Dialogue Action corpus is collected from Chinese teaching websites and is composed with dialogue with explicit topics, such as weather, transportation, etc. Since the conversations in this dataset are domain independent, we didn't need to do much preprocess work on it. However, this datasets is much smaller than the Jingdong corpus, it only consists of 3265 utterances. We also implemented agents to interact with Microsoft Chatbot XiaoIce in order to get its responses to expand our dialogue data and finally obtained nearly 8k utterances.

For music recommendation module, we crawled music data from music website wangyiyun[1], including the song name, the singer, the genres, the playlist it belongs to, the rank, the lyrics, the composer, and the comments from users. These meta data is organized as JSON format. We also downloaded the corresponding mp3 files in order to analyze the songs' audio features and provide listening links to those songs for users. The basic statistic information is displayed in Table 1.

Num of Songs	5994
Num of Artist	2676
Num of Labels	69
Avg num of Comments	11.485

Table 1: Statistics of song data

Although every song has a tag marking the genre it belongs to, we want to mining the hidden correlations between two songs through the lyrics themselves. We first conduct statistics on the tags of songs to explore the distribution of the songs' styles. Result shows that the style of the collected music has obvious long tail characteristics: the four tags of 'popular','Chiese','nostalgic','classic' account for more than

[1]https://music.163.com

80% of all songs, other tags like 'rap','relaxing' only take a small part of the dataset. We also used LDA (Latent Dirichlet Allocation) to analyze the lyrics of songs and attempted to evaluate the topic distribution of each song. The document-topic distribution vectors were then processed using PCA technique to reduce their dimensions to two to assess the effects of LDA. Figure 3 demonstrates the visualization results of LDA. Once obtained the topic distribution, we can evaluate the similarity between two songs according to their lyrics' topic distribution vectors.

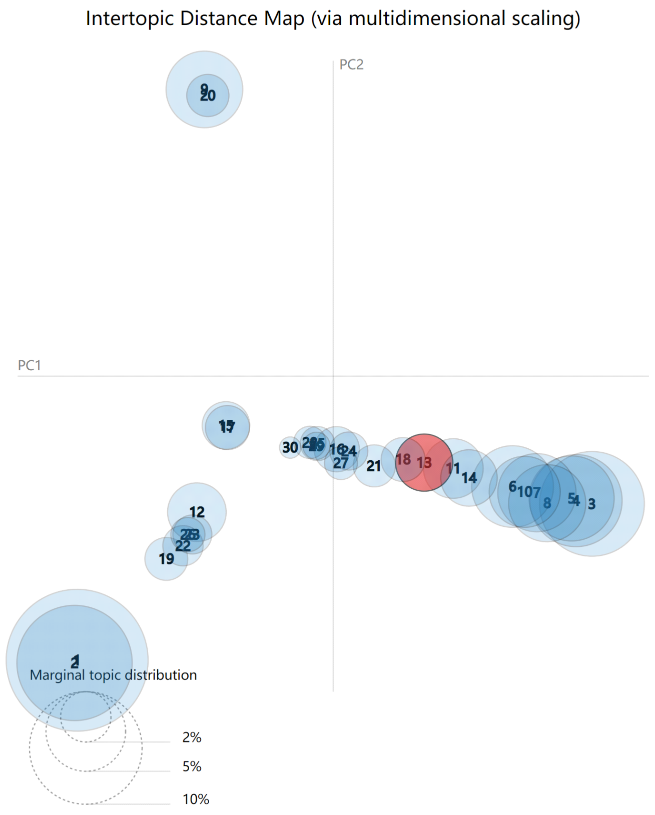


Figure 3: 20 topics distribution on lyrics of songs using LDA

3.2 Chat Module

The chat module was set to handle the single-turn conversation. We choose a **Seq2Seq** model to implemente this module. The input of the seq2seq model is a variable-length sequence, and it will also return a variable-length sequence as output. Figure 4 demonstrates the architecture of our chat module. The model composed of two parts. The Encoder part is a multi-layered Gated Recurrent Unit [Cho *et al.*, 2014]. We used a bidirectional variant of the GRU to take the advantage of encoding both past and future context. The embedding layer will map each word to a feature space and encode those words having similar meaning to close feature vectors. The hidden state of the encoder is calculated as

$$h = [h_1, h_2, ..., h_L],\ where\ h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}],$$
$$\overrightarrow{h_i} = GRU(\overrightarrow{h_{i-1}}, x_i), \overleftarrow{h_i} = GRU(\overleftarrow{h_{i-1}}, x_i)$$

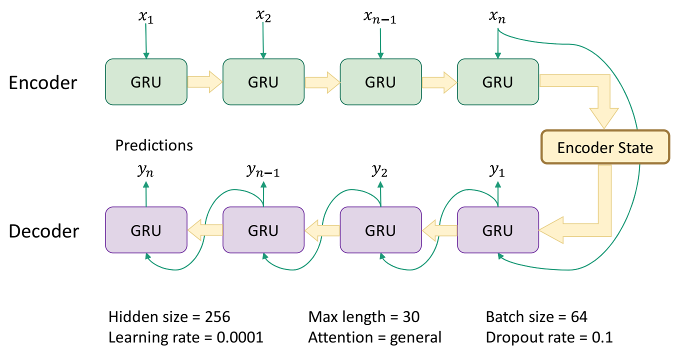


Figure 4: Architecture of the chat module

The decoder part is also a bidirectional GRU network. Since it is likely to loss information if the decoder only rely on the context vector to encode the entire sentence's meaning, we use attention mechanisms in this part to allow the decoder get weighted sum indicating which parts of the encoder output to pay attention to. The attention energies between the encoder output and decoder output are the so called "score functions", which can be calculated as

$$score(h_t, \overline{h_s}) = \begin{cases} h_t^T \overline{h}_s & dot \\ h_t^T W_a \overline{h}_s & general \\ v_a^T tanh(W_a[h_t; \overline{h}_s]) & concat \end{cases}$$

where h_t denotes the current target decoder state and $\overline{h}_s$ is all encoder states.

The parameters of the seq2seq model is initialized as Figure 4 shows, and it was trained for 200,000 epochs. Although dialogue generation is somewhat a subjective task, we still conducted experiments to assess the quality of the generated sentences. The evaluation results would be displayed in Section 4.

3.3 Music Intent Detection

The M_0 module is suppossed to judge whether an utterance from the user containing an intent of request to music or not. We adopted a convolutional neural network (CNN) to accomplish this binary classification task since it has been provde to be a powerful model to deal with sentiment analysis and question classification [Kim, 2014]. As shown in Figure 5, utterance of length n is represented as

$$x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n$$

where $\oplus$ denotes the concatenation operator, and x_i is the embedding of the ith word in the utterance. The filter $\mathbf{w}$ convolved with each possible window of size h to produce a feature map

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}]$$

where $c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$. Max pooling operation was then used to capture the most important feature over the feature map. The model used 128 filters to obtain 128 features. The features were then passed to a fully connected layer to calculate the probability distribution over labels.

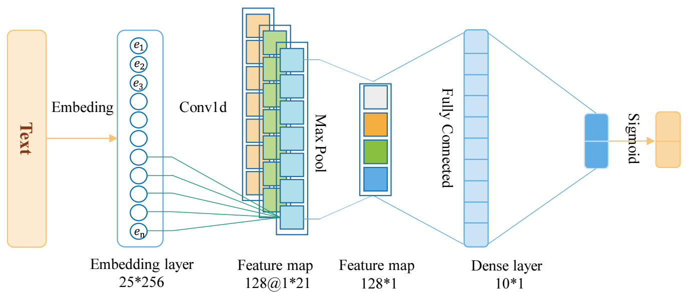


Figure 5: Architecture of the music intent detection module

By cause of the lack of real data for music intent detection, we decided to generate training data with some strategies. The dialogue training data for chat module, which are composed of utterances with no music intent, can be used as the negative cases. The positive cases can be gained by means of combining the music metadata and predefined templates. For example, as shown in Figure 6, we designed patterns for singer, the real singer names extracted from music data could be filled into those slots (represented as brackets), therefore one singer name could be used to generate serveral utterance with music intent. We also generated positive cases for other types of slots with same techniques, such as genres, song names, lyric writers and etc. Using this method, we finally obtained 210,394 positive cases and 232,046 cases. Since the templates can not contain all possible music-relative situation, the construction of training data is an iterative process: after the system was running, we invited volunteers to use it and collected the wrong classification cases to add them into the training set.

```
'有什么推荐的{}歌曲', '想听一些{}的歌', '{}的歌词是什么'
'来一首{}的', '找一首{}的歌', '要{}的歌', '{}的歌有吗'
'{}的歌有吗', '有{}的歌吗','有{}的歌吗','我想找主题是{}'
'我想找风格是{}的歌', '我想找标签是{}', '那首{}的歌'
'来一首style是{}', '找一首风格是{}的歌','找一首标签是{}的'
'找一首主题是{}的歌', '找一首style是{}','有什么好听的{}'
```

Figure 6: Positive case templates for music intent detection

3.4 Search & Passive Recommendation

When the user has an intent to inquire explicit information about music, the module M_1 will take the search task. Based on the metadata the songs have, we constructed different searching patterns. Once the query condition was triggered, the M_1 module will first analysis the types of slots in the utterance, then it will search in specific fields according to the constraints. The module uses Levenshtein distance to calculate the differences between the word sequences in order to implement a fuzzy search function[Levenshtein, 1966].

We mannually designed 27 search types and more than 100 natural language patterns in total. The 27 search types can be attributed to 4 main search patterns: 1) song-based search: users can search for basic information about a song, including the artist, the writer, the composer, the style, the lyrics, the comments and etc.; 2) artist-based search: users can search for all songs sang by some singer or created by some composer; 3) lyrics-based search: users can search for a song by giving a piece of lyrics; 4) style-based search: users can search for songs of an artist given the style.

The above-mentioned search types form the fundamental of passive recomendation module M_1. M_1 will utilize the information like other users' comments and likes to recommend most popular songs that satisfy users' needs. For example, if the user enjoys a song recommended by the system, he or she can ask the system for recommending other songs sang by the same singer thus to gain a group of similar songs.

3.5 Initiative Recommendation

It is a common situation that the user has a desire to listen to some music but he or she doesn't have a clear intent to select which kind of music to enjoy. In such case, the module M_0 will detect that the user's utterance contains music intent while the analysis of module M_1 and M_2 shows there's no slots has been filled, then the module M_3 will be invoked to undertake the initiative recommendation. The main idea of M_3 is to utilize "wisdom of crowds" and the user's personalized data to guess current user's preference.

The recommendation algorithm of M_3 is indeed a kind of collaborative filtering method but it has unique characteristics that can be distringuished from the common collaborative filtering algorithms. As demonstrated in Figure 7, both users and songs are treated as nodes in a undirected graph, where the blue circles denote songs' nodes while yellow circles denote users' nodes. If a user has had a comment on a song, its corresponding node will have a edge connecting to the song's node, which shows that the user is interested in and has listened to the song.

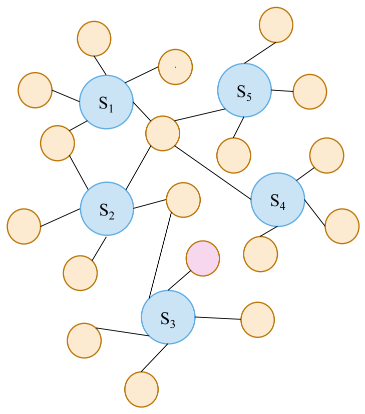


Figure 7: Song-User comment graph

In order to evaluate the similarity between users' taste, the link relationship should be mapped into continuous feature vector space. We employed **node2vec** to embed the nodes of the Song-User comment graph into vector space[Grover and Leskovec, 2016]. If the cosine distance of two vectors is close to 1, it means the corresponding users may have similar preference on music. When a new user starts to use the system, its node representation will be initialized as the average value of all the other nodes' representation. Then the user will give feedbacks to the system's recomendation: if the user shows interest to or give positive feedback on some song, then the mean value of all the other user nodes which link to this song will be used as the new representation for the current user's node; otherwise, if the user gives negative feedback on a song, its node will be pushed away from the song's adjacent nodes. The updating weight for negative feedback is set much larger than that of positive feedback because we suppose that dislike is a feeling stronger that liking.

4 Experiments and Results

The chat module M_4 trained for 200k epochs in total and saved its checkpoint every 10k epochs. We use a three-level evaluation indicator to assess the quality of the generated responses:

- 0: poor syntactic structure or poor semantic cohesion;
- 1: good syntactic structure or good semantic cohesion;
- 2: good syntactic structure and good semantic cohesion.

We sampled 200 query-response pairs from training set and external dataset respectively. Six volunteers evaluated the quality of responses by different checkpoint independently. The reliability of agreement of their scoring results was then assessed using Fleiss kappa indicator. It can be watched from Table 2 that the performances on each steps are close to each other and the 20,000 checkpoint has the best overall performance, the bad cases only take a little proportion (10%-20%). Table 3 displays the results when adopting external data as queries. In such case, the performance on each checkpoint are very different and the 200,000 checkpoint has the best overall performance, the bad cases account for a large proportion (nearly 50%).

	0(Bad)	1(Good)	2(Perfect)	Fleiss Kappa κ
10000	0.12	0.44	0.44	0.7261
20000	0.12	0.33	0.55	0.4745
30000	0.18	0.31	0.51	0.5301
40000	0.12	0.37	0.51	0.4223
50000	0.15	0.37	0.48	0.4002

Table 2: Training sample performance

	0(Bad)	1(Good)	2(Perfect)	Fleiss Kappa κ
10000	0.58	0.27	0.15	0.4515
20000	0.65	0.24	0.11	0.3728
50000	0.65	0.25	0.1	0.4199
100000	0.68	0.24	0.08	0.4524
200000	0.48	0.29	0.23	0.6289

Table 3: Training sample performance

[*] Fleiss Kappa κ: 0-0.2 (slight), 0.21-0.40 (fair), 0.41-0.6 (moderate), 0.61-0.8 (substantial), 0.8-1 (almost perfect)

From Table2 and Table3, we can see that the performances of the chat module on training data and test data are quite different, which shows the robustness of the chat module still needs to improve. Another possible reason for this phenomena is that the trainset is too small and needs to expand.

Table 4 shows the test results on music intent module M_0, the F1 scores on trainset and testset are both more than 0.96, which indicates the effectiveness of this module.

	Precision	Recall	F1
train set (309708)	0.99	0.99	0.99
test set (132732)	0.97	0.96	0.96

Table 4: Evaluation results on M_0

Finally, three volunteers were invited to evaluate our recomendation system. The score ranks from 0 to 3 for each utterance, where 3 means the response (music recommendation or chat) satisfies the user's expection while 0 is not. Each volunteer interacted to the system with 20 turns in total and evaluated it from both music-recommendation and chatting aspects. They finally also gave scores on the whole interaction process. Table 5 displays both the assessment on single utterance and the whole process.

	Average Satisfaction	Whole Satisfaction
Evaluator#1	1.6	1
Evaluator#2	1.8	2
Evaluator#3	2.7	3
Average Score	2.03	2

Table 5: Evaluation on whole system

From Table 5 we can see that the average satisfaction on a single utterance is relatively high while the satisfaction on the whole process is only 1, which indicates any bad recommendation in a single turn will have a big influence on the satisfaction on the whole process. A noteworthy point is that everyone has its individual dialogue pattern, which could be the reason that raises unsatisfaction since most volunteers' satisfaction raised a lot after we adjusting the system according to the new test cases.

The system calls the WeChat interface to provide the service. One can communicate with it through WeChat.

5 Conclusion

We designed and implemented a dialogue system for music recommendation, and accomplished most targets in our proposal. From this project, we not only learned a lot about recommendation systems and chatbots but also acquired a lot of experience of building a practical system. There would be many unexpected situations before we get our hands dirty, such as the lacking of corpora or unforeseen users' requirements. However, all these exceptions could not only make our system more robust but also stronger ourselves.

Due to time constraint, the current recommendation system is mostly based on text analysis and user preference, we hope it could deal with audio information and mining their relevance in the future.

6 Acknowledgement

I've learned a lot from my partner Jia Chen, she proposed lots of feasible plans and always accomplished the mission on time although she has a lot of work to do. I would like to take the chance to give thanks to her.

References

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[Hill *et al.*, 1995] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[Levenshtein, 1966] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[Soleymani *et al.*, 2015] M. Soleymani, A. Aljanaki, F. Wiering, and R. C. Veltkamp. Content-based music recommendation using underlying music preference structure. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, June 2015.

[Song *et al.*, 2012] Yading Song, Simon Dixon, and Marcus Pearce. A survey of music recommendation systems and future perspectives. 06 2012.