Unsupervised Context-Based Similarity Learning for Music Recommendation

Yue Yu (s1563228)

Abstract
The mathematical nature of music, that is each song can be characterized by some vectors, has been widely used for music recommendation systems. However, the social nature of music has comparatively received less attention. There are some patterns from the music listening context of various users can be learned to construct the relationship graph of music. In other word, based on enough data, the characteristics information within thousands of millions pieces of music can be exposed by people's behaviors during listening to music. This paper proposed an approach based on the music listening sequence collected from social music networks, like Last.fm, to predict interesting songs for the user. This method can be implemented without any knowledge about the music itself.

Keywords
Context-Based — Similarity Learning — Recommendation

*__Team members__: Yi Huang, Yue Yu

Contents

Introduction

Problem Statement Music recommendation systems is an important topic for both research and business. However, as the Internet goes bigger and bigger, there are many audios can not be identified or some essential information may be missing. In this situation, unsupervised music recommendation algorithm can be a life saver.

Related Work Last.fm dataset is a wildly used API for Social Network, Machine Learning and many other research areas. There have a lot of research done based on this dataset. For example, some authors used it to predict future links between users [1][2].

1. Methodology

In this project, we first preprocessed the dataset to extract the sequential information of music which is the order they were played by each user. Based on this we can calculate the strength of relationships between any two songs with the Pointwise Mutual Information (PMI) function [3] as explained later. After that, we can already predict songs that users might be interested in by recommending music that is closely related to the songs users have listened. Moreover, Markov Cluster Process (MCL) [4] was introduced to detect the music communities based on the similarities of music pairs. This method was used before prediction in order to further improve the performance of this recommendation algorithm.

1.1 Data Preprocessing

Since the performance of this prediction algorithm is greatly relied on the volume of user track information (i.e. playing sequence of music), we used the Last.fm Dataset - 1K users dataset [5]. This dataset contains (user, timestamp, artist, song) tuples collected from the Last.fm (www.lastfm.com) API, representing the listening history (until May 5th, 2009) for 992 users.

We have preprocessed this dataset using the following steps:

1. Most songs (88.67%)[6] in this dataset are identified with a unique MusicBrainz Identifier. For the music that do not have one, we created unique identifiers for them based on their title and artist.
2. Songs played less than 5 times in the same user's track are removed to exclude the songs that users may not be

very interested in and reduce the dimensions of music relationship matrix.
3. The most recent 30% of each user's listening track were left aside for testing, and the remaining data was used for training.

The implementation details can be found in Yi's report[6].

1.2 Similarity Analysis

In this implementation, the similarity between any two pieces of musics is defined by Pointwise Mutual Information (PMI) as the followed equation. PMI was firstly suggested by Turney[7] as an unsupervised measure of the semantic similarity between two words. Just like words, the music played together in a users track list can be either random, for example after spent much of the afternoon listening to pops the user happened to play a classical music, or related to the context, for instance users may have some playlists with similar music in each of them. Therefore we can compare the expected probability of songs were randomly played with the actual probability of pairs of songs played together to tell how the songs are related.

$$\text{pmi}(x;y) \equiv \log_2 \frac{p(x,y)}{p(x)p(y)} \tag{1}$$

The co-occurrence probability of pairs of songs $p(x,y)$ is defined based the music shows in a distance window of 10 on the users' playing track. That is if music x was played within next 10 songs played after music y, we say music x and y co-occurred once. The occurrence probabilities of music x and y which is $p(x)$ and $p(y)$ are calculated based on their occurrences in all the windows.

The music relationship matrix we finally got through this method is a $1,500,646 * 1,500,646$ matrix with 36,377,070 non-zero relations. For better time (i.e. looking up) and space efficiency, we stored it as a sparse matrix in Compressed Sparse Row format.

To verify the performance of PMI function on the Last.fm dataset, we randomly chose some music and checked whether the PMI values reveal some characteristics of music in the real world. Table 1 shows the seven most related songs founded for "From Now On" by Supertramp.

Table 1. Related Songs of "From Now On"

	Features	
Title	Artist	Genre
Hide In Your Shell	Supertramp	Pr[1]
Cannonball	Supertramp	Pr[1]
Rudy	Supertramp	Pr[1]
Hold Me, Thrill Me, Kiss Me	Gloria Estefan	Pop
Divine Invasion	Trans Am	Ar[2]
Dance For Me Baby	Slaughter	Pop
One Bad Apple	The Osmonds	Pop, R&B

[1] [2]

1.3 Music Community Detection

With the connection weights (i.e. similarity) between songs, we constructed a weighted network of music. Then, we applied the Markov Cluster Algorithm (MCL) to find music communities within the graph. MCL was used because of it's efficiency and scalability. I have tried several clustering methods for this project including MCL, SLPA and OSLOM. MCL is the only one of them can handle our $1,500,646 * 1,500,646$ sparse matrix in reasonable time. The time complexity of MCL is O(N k2), where N is the number of nodes in the graph, and k is the number of resources allocated per node which can be chosen surprisingly low without affecting clustering quality [4].

One of the obvious drawbacks of MCL is that it cannot generate overlapping clusters very easily, since it requires some particular type of symmetry to be present in the input graph[4]. But we can live with that for now.

1.4 Prediction and Testing

The Story Without Clustering We first chose top N frequent songs (Voters) from each users' training dataset along with their occurrence. Then, we located K most related songs (Search Beam) for each of the N frequent songs along with their similarities (PMI values). After that, each Voter voted for their K most related songs based on the strength of their relations and the Voter itself's weight (i.e. occurrence in the same user's track list). The duplicated candidates were merged, for example, both music A and B voted for music C and their votes would be added. Finally, we calculated the voting results for up to $K * N$ candidates and recommend the top X songs to the user.

The Story With Clustering The first step is the same as above. Chose top N frequent songs from each users' training dataset along with their occurrence. Then, we located K most related songs **from each clusters where the N frequent songs lied in**. The things after that are also the same as above.

The testing process is relatively simple. The recommended songs were check one by one whether they occurred in the user's testing dataset which is the most recent 30% music the user have listened.

Followed by the parameter settings in this project:

Table 2. Parameter Configuration

Voters(N)	Search Beam(K)	Recommendations(X)
20	[1,2,3,4,5]	20

[1] Pr refers to Progressive rock, the term was roughly interchangeable with rock music, referring to "progressive" pop music created for listening attentively [8].

[2] Ar refers to Alternative rock.

2. Results and Discussion

2.1 Accuracy Analysis

Table 3. Prediction Accuracy

Search Beam	With Clustering (%)	Without Clustering (%)
1	72.94	67.43
2	65.23	59.42
3	62.52	55.75
4	61.30	54.52
5	60.59	54.61

As Table 3 above shows, the prediction algorithm implemented based on clustering result (community detection) works achieved better accuracy compared with the prediction algorithm only based on music relationship matrix.

However, as the search beam increases, the accuracy of both algorithms shrinks obviously which is beyond our expectation. It seems that the more related songs were searched, the less effectively the algorithms will perform. One of the reason could be that, because of the limitation of dataset (only 1K users were included), most of the songs distribute sparsely over the music space and smaller search beam will lead the algorithm to find more songs that have been heard by the users both in the training and testing dataset. As Yi suggested that after deleting the songs which that user has listen to before, most users' prediction accuracy is close to 0 [6]. According to the following two figures, it is clearly that most of the songs have very few neighbors and are lying in very small clusters. This problem could be relieved by introducing bigger dataset with more user variety.

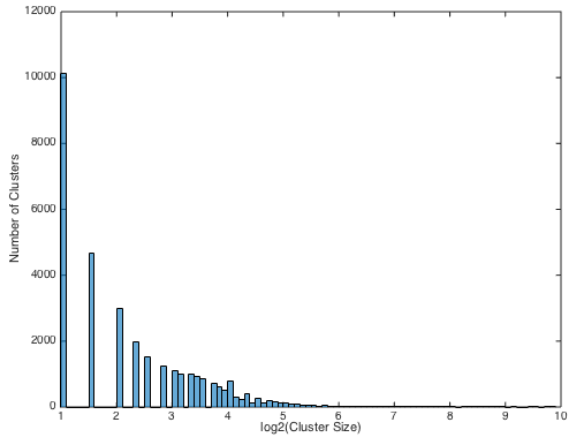

Figure 1. Distribution of Clusters

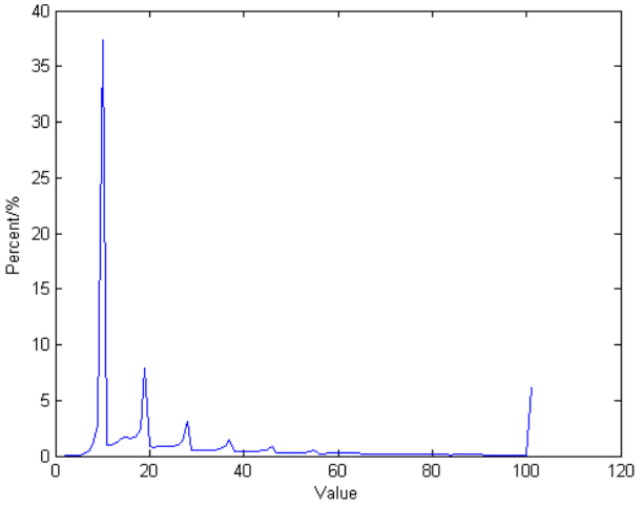

Figure 2. Distribution of Neighborhood [6]

2.2 Efficiency Analysis

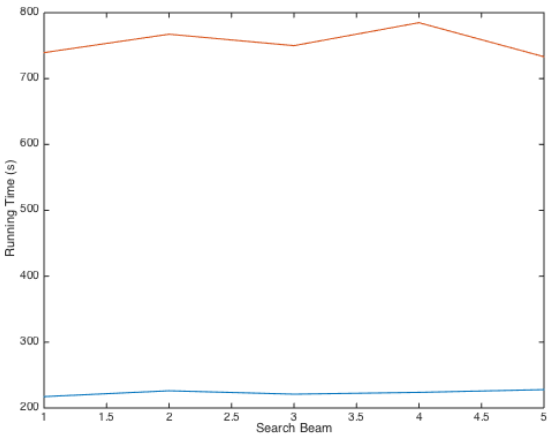

Figure 3. Distribution of Neighborhood [6]

As figure 3 above shows, the prediction algorithm implemented based on clustering result (blue line) works significantly faster than the prediction algorithm only based on music relationship matrix (orange line). This matches our expectation, since the blue algorithm shifted some of the queries from the music relationship matrix to the clustering table. The size of clustering table, which records the music id consisted in each cluster, is much smaller compared with music relationship matrix with all the music pairs. Therefore, it takes much less time to query for related songs from the clustering table compared with music relationship matrix.

In this project, the size of clustering table is 2.4MB and the size of music relationship matrix is 394.2MB.

References

[1] Xu Feng, JC Zhao, and Ke Xu. Link prediction in complex networks: a clustering perspective. *The European Physical Journal B*, 85(1):1–9, 2012.

[2] Neeral Beladia and Neera Vats. Influence based link prediction using supervised learning.

[3] Pradeep Muthukrishnan. *Unsupervised Graph-Based Similarity Learning Using Heterogeneous Features*. PhD thesis, The University of Michigan, 2011.

[4] Stijn Van Dongen. Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications*, 30(1):121–141, 2008.

[5] O. Celma. Music recommendation datasets for research. `http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html/`, 2010. [Online; accessed 19-Nov-2015].

[6] Y. Huang. Context-based user behavior prediction. *Course Project for Social and Technological Networks*, 2015.

[7] Peter Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. 2001.

[8] John Shepherd. *Whose music?: A sociology of musical languages*. Transaction Publishers, 1977.