Internet Radio Recommendation System Based on Sentiment Analysis

Congcong Chen
Texas A&M University
chencc6566193@gmail.com

Shen Ge
Texas A&M University
taoyy19@gmail.com

Xinru Ma
Texas A&M University
xinruma9111@gmail.com

ABSTRACT

The online radio station nowadays involves to have a recommendation system, but the user still cannot choose to listen songs that will fit their mood at the moment. So we develop a online radio station that support both the recommendation system and the mood selection function. So that the user can enjoy the songs that are recommended to them based on their favor and their mood! We use sentiment analysis[1] to determine the mood of the song, $K - means$ to determine the user clusters, and collaborative filtering to prepare scores for the final recommendation.

Keywords: Sentiment Analysis, Recommendation System, Mood Selection, Online Radio Station

1. INTRODUCTION

Do you sometimes find the existing radio stations are terrible when you have a bad day and feel frustrated? Are there always some songs that don't suit your mood at that moment quite well? You think you have to learn to lower your expectation of what the radio station can do, but the truth is you don't have to! We propose a new method to enable the radio station to provide you your favorite songs according to your mood! You tell the radio station your feeling right now, and it provides you with the right songs you need.

In this paper, we first build up a music recommendation system, we propose the idea of adding the mood factor into it by analysing its mood through the lyric analysis[2], and then we implement this idea into the system.

2. IMPLEMENTATION OF WORK

We have built a music radio station which has its recommendation system together with mood selection. This radio station consists of the following components. The Figure 1 is a high-level design figure that shows the connections among these components.

2.1 User Interface

Users can use this user_interface to login the system and interact with the systems. They could select/change their current mood on the mood window; they could also click on like or unlike to indicate their attitude towards current song. And this interface should display recommanded songs information to users and play this song. (have not realized playing songs fucntion due to lacking of audio documents)

2.1.1 *Front End Main Frame Layout*

The user interface consists of a main windows form. This form is displayed when the app starts to run. As shown in Figure 2.

This user interface program consists of several functions. And these functions could be roughly classified into two groups: 'initialization fucntions' and 'component event handlers'. And we will initialize the whole system at start time by calling these 'initialization fucntions'.We first initialize the main window by calling: Tkinter.Tk.__init__. Then we can initialize all the components of this user interface by calling self.initialize() and self.initialize_timer() function.After we have initialized all the components, the whole system wilal be looping infinitely waiting for the components events. As long as one components event happens, the cpu will execute corresponding event handler. And realize specific function. Detailed discription of the components:

1. PhotoImage: used to read picture_name.gif files and create a photo object then the 'canvas' component could display this picture using this photo object.
2. Label: used to display words onto the window form.
3. Scale: mainly used as a progress bar to indicated the current process of the playing procedure, can also be used as a volume adjusting bar.
4. Button: used as various functional button, such as 'love', 'skip', 'login' as so on.
5. Radiobutton: used to let the user to select their current user mood.
6. Toplevel: used to pop up a new window form which is used to ask user to enter their user ID.

2.1.2 *Back End Data Processing*

Because User Interface Program is the entry of the whole program. So this program would import several other packages (written by other group memebers) and use several

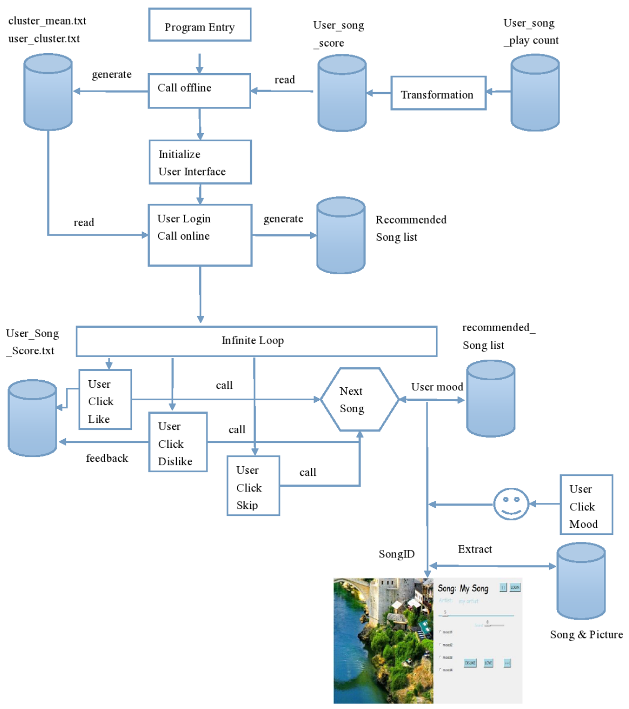

Figure 1: Connections between main components of online radio recommendation system

other processed datasets (generated by other group memebers) during executation process. The packages are: 'offline.py', 'online.py'.The datasets are: 'user_movie_dict.txt', 'songs_list.txt', 'song_score.txt', 'map_songid_songname'. The main processing procedure is:

1. When initializing, program would call the function of offline.process_data_main_entery().
2. After initializing, the program enters infinite looping state and waiting for widget event.
3. At this time, user should login with UID. Then, the program would call online.recommand_Song() And get returned a sorted recommand song list.
4. Then the program start to select and play song from the above returned song list.
5. Each time a specific song is slected, program will compare the mood of this song (read from song_score.txt file) with the user mood, if they are match, namely the same, then we will play this song, if not, we will select the next song from the above returned song list and continue the above process until we select one song that match user's current mood.

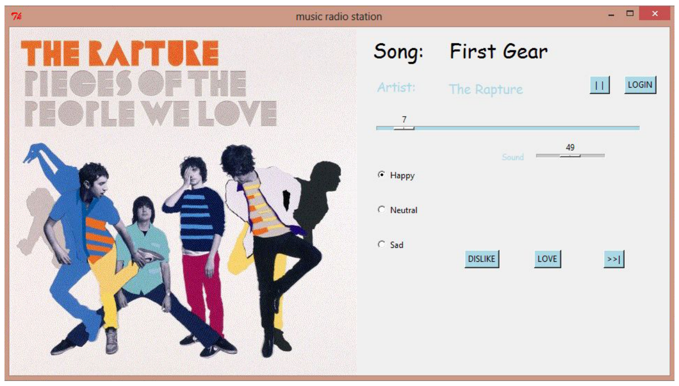

Figure 2: The song playing is 'First Gear' by 'The Rapture', the user mood is 'Happy'

6. User could also click 'LOVE', 'DISLIKE', 'Pause' to control the song playing process. When user click 'LOVE' or 'DISLIKE', program will record the behavior of user by adjusting the song score of this user.

   For 'LOVE', the fomular is as follows:

   $$new\_score = old \times 0.5 + 0.5 \times 5.$$

   For 'DISLIKE', the fomular is as follows:

   $$new\_score = old \times 0.5 + 0.5 \times 0.$$

   When the system exits, the program would write this update score to user_movie_dict.txt. And in this way, we provide a feedback mechanism. So the next time, user login, the recommanding algorithm would run on this new dataset and recommand songs accroding to this new dataset.

2.2 Music Database

All the data of our project is from the Million Song Dataset (MSD), a dataset which have stored a million of songs and a great amount of related information. We retrieved three files from this dataset: 'train_triplets.txt', 'mxm_dataset.txt' and 'unique_tracks.txt'.

The 'train_triplets.txt' records the listening history of all the users, namely the listening frequency of every user to every song. This file can help to calculate the utility matrix used in the user_clustering_component. To improve the performance of the clustering algorithm in the following step, we select 100 users who has listened to at least 100 songs out of the total 5574 songs in the dataset. The 'mxm_dataset.txt' records the lyrics of the 5574 songs. The lyrics are stored as stemmed words in a dictionary. To analysis the mood of the lyrics, we used Mashape API to realize the sentiment analysis[3][4]. Every time we sent the lyric of a song to the API, the API can return a mood lable ('neg', 'pos' or 'neutral').

The 'unique_tracks.txt' records the title, singer, songid of each song. The titles and singers of the recommended songs will show up in the user interface. Songid is used to match a particular song in 'train_triplets.txt' and 'mxm_dataset.txt'.

2.3 User Collaborative Filtering Component

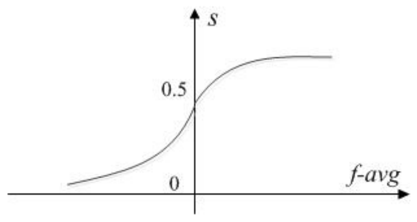

Figure 3: The trend of score function

First, we build a utility matrix where each user is represented by a feature vector which records the scores he or she gives to every song. The score for each song of each user is calculated by the following formula using the listening history of the user, where s represents the score, f represents the times the user has listened to a particular song, and avg represents the ratio of the times the user has listened to all the songs through the radio station over the number of songs he has listened to through the radio station.

$$s = \frac{1}{1 + e^{-(f-avg)}}$$

The trend of the formula is shown in Figure 3. We assume that if s is greater than 0.5, the user likes the song, and these songs will have higher probability to be recommended to the user. Then, after we got the utility matrix, we found that the matrix is very sparse, hence it will be very hard to detect similarity among users. For example, it possible that two users have very similar taste for music but have not listened to any songs in common. Therefore, we decided to use $K-means$ algorithm to cluster the users before user collaborative filtering. We clustered these users into several groups using $K-means$ clustering method. After we have user clusters, the songs selection for a specific user is based on his/her favorite songs and the specific cluster he/she belongs to.

After we got K user clusters, we used user Collaborative Filtering to realize the recommendation of the songs. More specifically, to calculate the feature vector of a cluster, we first use $K-means$ method and take the mean of the vectors belong to a cluster as the cluster's feature vector. However, there may be many songs that all the users in the cluster have not listened to, therefore many elements of the feature vector would be zero. To predict the scores of these songs that has never been listened to be the user, we use Collaborative Filtering. The Collaborative Filtering consists of three steps:

1. Consider cluster c;
2. Find set D of other user clusters whose scores are 'similar' to c's scores;
3. Estimate c's scores based on scores of clusters in D. Finally, the candidate songs for recommendation will consist of two parts. The first part are the old songs which are considered to be liked by the user according to the listening history. Another part are the new songs which have high scores in the feature vector of the cluster the user belongs to.

2.4 Mood Classification Component

After the user_clustering_componet, we have a candidate set of songs to be recommended. Before we send our songs to the user, we have to determine whether or not they fit for the mood of the user at that time. We first preprocessing the songs in our database, use sentiment analysis methods to process the lyric, and assign each song a mood label ('happy', 'sad', 'neutral') according to the sentiment analysis results. There are also three mood choices in the user_interface, namely 'happy', 'sad' and 'neutral'. Finally, we select songs whose labels match the user's choice from the candidate set, and recommend these songs to the user.

3. RESULTS

The original code and datasets have been zipped into the file code_and_data.rar. Our project has achieved the three goals we have posted in the proposal. First, we have realized the lyric sentiment analysis of the songs (5547) in the dataset. Second, we have successfully used $K-means$ user clustering and user Collaborative Filtering to build the recommendation system. Finally, we have built up a user interface via python as is shown in Figure 2. Through the user interface, the user can get the recommended songs that match his listening history and current mood. If the songs have been downloaded in the computer, the user interface can even play the songs.

To make a specific example of the successful of our system, we retrieved the lyric of one song as shown in Figure 2 and we listened to the song in Youtube on the following link: https://www.youtube.com/watch?v=1-2Km3VqEi4. From many aspects as of lyric, beat, rythm, title, intention of this song, we concluded its a happy song just as the user requires in the mood of 'Happy'.

4. FUTURE WORK

We think there are two aspects about this project that can be further explored in the future. First, we only consider the lyric sentiment analysis in our project, which is not enough for song recommendation system. It would be better if more features of the songs can be taken into consideration to determine their mood, such as Genre, Album, Year and Beat. We may use machine learning methods to determin the final coefficient of each feature. The second part is about the evaluation of the system. We think that the best way to evaluate a recommendation system is users' feedback on the recommended things. We can record a user's feedbacks on all the recommended songs the user has listened to. If the user 'likes'(through the 'like' and 'dislike' button) 40% of the recommeded songs, the system should be considered to be successful.

5. REFERENCES

[1] Liu, B., Sentiment analysis and subjectivity. *Handbook of Natural Language Processing.* N. Indurkhya and F.J. Damerau, eds. 2010.

[2] Liu, B., Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies.* Morgan & Claypool Publishers, 2012.

[3] Taboada, M., J. Brooke, J., Tofiloski, M., Voll, K. and Stede, M. Lexicon-based methods for sentiment

analysis. *Computational Linguistics 37*, 2 (2011), 267-307.

[4] Pang, B. and Lee, L. Opinion mining and sentiment analysis. *oundations and Trends in Information Retrieval 2*, 1-2 (2008), 1-135.