Final Project Progress Report

Report two

Xingbang Liu, Matt Jones, Travis Thomas

April 14, 2018

So far, we have finished implementing the basic structure of the program. Our main class is a do while loop, if the user doesn't want to quit, the program will continue by load conversation class. Conversation class is a class that will require commands from user. Based on different commands, the class will redirect the user to different classes. For example:

```
Menu: 1.Like | 2.Explore | 3.Daily Selection | 4.quit
```

If user types "1", then the program will direct the user to liked song list, then like class will show user the music list. The like class will also ask the user how to sort the list. The rest of the important classes, such as ***explore()*** and ***dailySelection()***, will implement our random suggesting system and forecast suggesting system.

The explore class reads the Cloud music list CSV file and randomly selects a few songs and feeds it to user, based on song's name initial letter sequence. If the user likes any of the songs, we will ask if the user want to add it to his or her liked music list. If they do, then the program will mark a song with y in the last column. and add it to like list. Because of the nature of how we mark liked songs, the program will randomly select a list song from our list, containing songs which the user has liked and not-liked. This way, the program is generating a list exploring songs outside of the users "liked songs" while still featuring some other users liked songs. This code will use the bufferedreader tool and previous lab code to read in information, and write a y in the last column of our file when needed. Finally, the search algorithm will search through the cloud list, find the song that user wanted, then copy the song to like list.

In the daily selection class, the program will select songs that the user may like based on the forecasting of whether the user will like it (The algorithm will be explained later). The whole idea of this algorithm, is that it uses data from the users liked songs to try to figure out what other songs the user would like (see algorithms section for more information). Then, a dailySelection of songs is created for the user that they can listen to that day, featuring songs that they know (from their liked list) and new songs.

Here is the newest structure we have:

```
main()
  do conversation();
  while answer()

conversation()
  if(1)
    likes();
      if answer()
      sortList();
  else if(2)
    explore();
  else if(3)
    dailySelection();
  else if(4)
    break;
  else
    prumpt
```

```
likes()
  print like list(csv);

explore()
  print(random select 5 songs);
  if answer()
  addSong;

addSong()
  write(csv)
  //Search algorithm(not implemented yet)

dailySelection()
  //not implemented yet

answer()
  if(y)
  true;
  else if(n)
  false;
  else
  answer();

sortList()
  sortInt();
  sortString();
```

Algorithm Picking

For the algorithm for forecasting potential user like song, which is Collaborative filtering, there are two algorithms that we know.

Memory-based

The first one is user grouping method. In this method, the algorithm will first track down people who have similar activities, then recommend songs to each other by their playlists. For example, scores are assigned to different actions:

```
Repeat song = 5, Share = 4, Like = 3, Play = 2,
Played completely = 1, Skip = -1, Dislike = -5
```

Then, a persons preference would be a N-Dimensional vector. Here N is the number of songs by default. By using the cosine of the vector angle, which is generated by two vector, we can know how similar two users can be. The cosine of 0 degree, which means two people are exactly the same, is 1. The cosine of 180 degree, which means two people have the opposite preference, is -1. The equation is:

Assume we have vector $\vec{a}$ and $\vec{b}$,

$$cos(A) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \times |\vec{b}|}$$

This prediction method is very accurate, but there are too many calculations and it is not accurate for a new user.

Model-based

As for the second method, the algorithm will analyze the playlist of a person, then do recommendations. Models such as the latent factor are used in this method. In latent factor model, user's like list will be analyzed by dividing into different tag factors, for example, classic and indie. By signing scores for different factor, we can have a matrix which can divided into two different matrix:

$$R = QP^T$$

$$R_{user,song} = Q_{user,factor} P^T_{factor,song}$$

$$= \begin{pmatrix} q_{user_1,factor_1} & q_{user_1,factor_2} & \cdots & q_{user_1,factor_n} \\ q_{user_2,factor_1} & q_{user_2,factor_2} & \cdots & q_{user_2,factor_n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{user_n,factor_1} & q_{user_n,factor_2} & \cdots & q_{user_n,factor_n} \end{pmatrix}$$

$$\begin{pmatrix} p_{factor_1,song_1} & p_{factor_1,song_2} & \cdots & p_{factor_1,song_n} \\ p_{factor_2,song_1} & p_{factor_2,song_2} & \cdots & p_{factor_2,song_n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{factor_n,song_1} & p_{factor_n,song_2} & \cdots & p_{factor_n,song_n} \end{pmatrix}^T$$

From the matrix R, we can get estimated score. After that, we can ignore the song which the user already listened, and recommend him the song he has not listened.

Of course, in the real world, if we implement algorithm based on song, the second dimension factor will rapidly increase. Plus there will be too many songs. Therefore, it will be more efficient if we calculate based on tags of the song. For example, for "The Sound of Silence" by Simon & Garfunel, tags can be Folk Rock, 60s, and Movie Track. Due to the time we have is limited, we would not build tags databases, therefore we would analyze by songs. Additionally, we also would not use the second method to calculated scores because we do not have time to build multiple users.

Challanges

In this project, the most difficult part is to implement efficient searching and suggestion algorithm. Also, it is very time consuming to build multi-user model. Therefore, we will be implementing a very simple and basic recommendation system. We would assume the potential scores are already calculated. We can implement the complete algorithm in the future.