Q3b System Design Document

1. Song Recommendation System

We propose a prototype for providing personalized song recommendation for all female users. The recommendation is based on the data which contains the users' historical information on their plays of songs and the basic users' information.

2. Goal

The song recommendation system provides the number of recommended artists' songs for an individual user. In general, the recommendation can be given according to the user's gender, age and country.

3. Design Architecture

This recommendation system is a data driven system. The database contains the historical information of each user (user id) on their played songs (with artist id and name) and the number of plays on their artists' songs [1]. The other database consists of user basic information such as user id, age, country and signup date [1]. The first database, which contains the user historical information of songs, is converted to a required rating matrix, where the rating matrix contains the rating information based on the number of plays on their songs. The second database, with the user basic information, is converted to a user feature matrix which contains the weight of the features.

The recommendation engine employs an open-source package lightFM which is operated in python. This recommendation engine is a hybrid engine of collaborative and content-based filtering. By supplying the required rating matrix and user-feature matrix to the engine, a number of top recommended songs can be given to each user.

The schematic of design architecture is provided a follows:

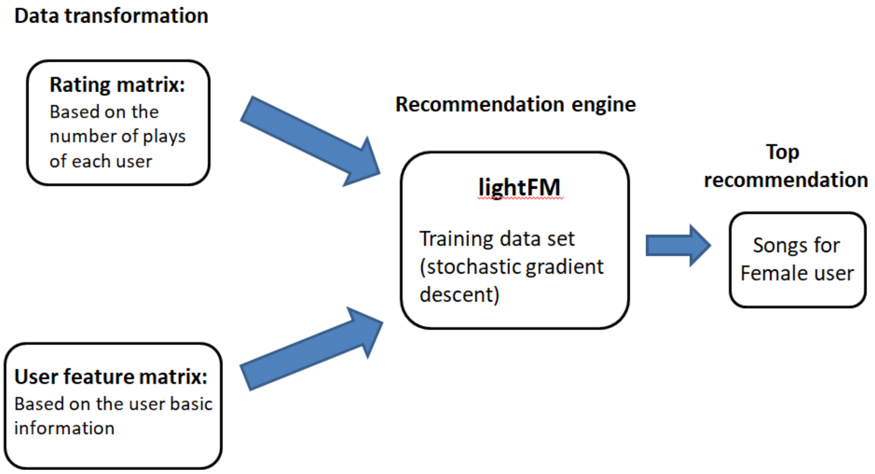
Data transformation
Rating matrix: Based on the number of plays of each user
User feature matrix: Based on the user basic information
Recommendation engine
lightFM
Training data set (stochastic gradient descent)
Top recommendation
Songs for Female user

Fig. 1: Schematic of design architecture.

4. Data Transformation

It is required to preprocess the datasets. The first database contains the number of plays on the songs of each user. Based on this information, the rating of each song for a user can be obtained by normalizing the total number of plays of that user. The effective rating matrix contains the rating of songs for all users, each row provides the rating of the songs of each user. Therefore, the sum of ratings in each row is one. The second database is transformed into a user-feature matrix.

5. Recommender engine

We adopt the open-source package, lightFM for recommendation. This provides a hybrid of collaborative filtering and content-based filtering. It is required to input the rating and user-feature matrices. There are number of parameters for training the dataset based on the stochastic gradient descent such as the learning rate, the number of epochs, the loss functions and learning schedules. The details can be referred to the documentation [1].

6. Environmental setup

The data preprocessing and transformation works on the python 3 by using python pandas, numpy and scipy. The lightFM package is implemented in python. We have demonstrated an example to implement of song recommendation by using jupyter notebook in python 3.

The required python packages are provided in the following:

1. pandas
2. scipy
3. numpy
4. lightfm

7. Major concern and limitations

The major concern of this song recommendation system in production is the scalability of the system. In the real application, the database size would be much larger. The computation effort in data preprocessing will become more expensive. The appropriate parallelization is required in this stage. Currently, the main tool for data cleaning and transformation is python pandas package. There is a suitable tool, which is the package Dask with built-in parallelization, for dealing with a bigger data size.

Second, it is required to train the data set for recommendation by using lightFM. When the data set becomes larger, the training time will increase significantly. The explicit scaling of training time and data size is required to investigate. The package lightFM provides a multi-thread parallelization (complied with open-mp) which would speed up the training process. However, the current version does not provide the computation support by using gpu. The further speedup would be limited.

References:

1. The information of song dataset: http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm360K.html
2. The document of lightFM package is: https://lyst.github.io/lightfm/docs/index.html