Song Recommendation User Guide

December 16, 2016

Team members: Jingyi Su (js5991), Luyu Jin (lj1035), Qianyu Cheng (qc510)

1 Introduction

Welcome. This is the user guide for song recommendation project. Thank you for playing.

This project applies song information from Million Song Dataset to devise a system that recommends playlists of songs to users. It uses decision tree model to recommend songs to users based on their selected preference for other songs. To get the recommendations playlist for songs, users can run script: **python SongRecommendation.py** to begin your song recommendation journey. But before that, let us briefly explain to you how song recommendation in our project work.

The code in project is consisted of two parts. First one is DataCleaning, and the second one is SongRecommendation. In this user guide, we will also explain how this program works using specific steps.

2 Data Cleaning

For DataCleaning repository, the main module that runs data cleaning is called **DataCleaning.py**. In the package inside DataCleaning repository, we extracted the data(in .json and .h5 formats) downloaded from the millionsong data set: http://labrosa.ee.columbia.edu/millionsong/ to data frames that are easier to use in further analysis.

The main data sets we used are: song-level tags related 'Last.fm dataset' and 'million song dataset', which contain information about songs and tags information for songs. Since the original data set is too massive, we didn't include them in folder. However, if needed the original data could be provided per request. In dataset, the columns are artists name, titles of the songs, year, and tags that are about content of songs, such as hip-hop, soul or pop.

Inside package, there are **datapreprocessing.py** and **test.py**.

datapreprocessing.py takes the raw data downloaded from millionsong website and converted all data files in h5 format and json format into a merged pandas DataFrame. It also performs several data cleaning functions, including removing unnecessary columns, dropping and filling missing values, deleting songs with no tags.

test.py is just a unit test for all the functions inside datapreprocessing.py. In order to successfully pass all test, please put MillionSongSubset folder and lastfm_subset folder in the same layer as the project.py file. Otherwise, you need to manually change the path of the data files below. Also, we only use a small portion of all data files to run the unit test. Otherwise, it will take hours. The result of the unit test is attached in folder as well.

After DataCleaning, the next step is to run some model on the cleaned data set (stored in cleanedData.csv). The cleaned data set is also massive, so it is not included in the folder as well.

3 SongRecommendation

For SongRecommendation repository, the main module that runs song recommendation is called **SongRecommendation.py.** In the package named song inside SongRecommendation repository, there are two python modules: **featureSelection.py** and **model.py**.

What **featureSelection.py** does is to select the features that are useful for song recommendation. It first takes in the cleansed data set we got after dataCleaning but without feature selection, then removes tag columns that have less than threshold songs associated with them. In the end, it saves Data after feature selection into a csv file.

For **model.py**, what this module does is that it first sorts the song data with song_hotttness, so that the first 20 recommended song are the most popular ones. As a result, the users of the recommendation program will know the first several songs the module recommends. Otherwise, if the users are not familiar with the song and artist that the model recommends, he or she could search on YouTube to know what song it is. However, if the user like or dislike all the songs, the decision tree algorithm won't work, so the module will only retrain the model when users have entered both like and dislike during the program. The target Value of the song would be from the user input: which is 0 or 1. Then we will use target value of the song as Y train data for decision tree, and using the decision tree model to apply to all the other songs and to predict their Y. Then, by ranking all the Ys for songs, we find the top one song and recommend it to users, and then retrain the model again. After the users are satisfied with the result, they could enter 'quit' in commend line, and a plot of feature importance will show up and be saved in current directory.

myexception.py includes user defined exception.

test.py is just a unit test for all the functions.

Hope that everything is not that overwhelming. If it is, don't worry. This song recommendation program is easy to use for every music fan with or without statistics or programming background. Just follow these steps below and you are good to go.

4 Steps for song recommendation project

The steps are:

1. Download music recommendation project from gitHub.

Log into gitHub. Use git clone or download to get the music recommendation project.

2. Open downloaded folder, and open terminal in folder SongRecommendation

Open terminal in folder and make sure that current path is '../SongRecommendation'

3. In the terminal, run with the code 'python SongRecommendation.py'

The song recommendation program is working right now.

4. Read instruction carefully. Look at the first song and enter '0' for dislike and '1' for like. Repeat for following songs. Initial recommendation would be 20 songs.

However if the users like or dislike all the first 20 songs, the decision tree algorithm won't work, so the module will only retrain the model when users have entered both like and dislike during the program. Thus, if the users keep liking or disliking all the songs, the system won't stop recommending after 20 songs until the users start to like or dislike certain song.

5. When you are happy with your result and want to quit, enter 'quit' and end the program.

A plot of feature importance will show up when enter ‘quit’. The module will ask whether or not the user want to save the plot or not, and plot will be saved in current directory. However, if the users only enter ‘0’ or enter ‘1’ all the time, the decision tree model won’t work. As a result, the plot won’t show up since no feature importances are calculated. The features are chosen from dataset, such as tags, title or artist or etc.

5 Appendix 1: Python libraries used in this project

We primarily used **Unit Test**, **Pandas**, **Numpy**, and **matplotlib**. We also used **sklearn**, **json**, **codecs**, **os**, **sys**.

6 Appendix 2: File directory

- DataCleaning
  - Package
    - __init__.py
    - datapreprocessing.py
    - test.py
    - test_data.csv
    - test_data_cleaned.csv
    - Unit Test Results
      - Test Results - Nosetests_in_test.html
  - DataCleaning.py
- SongRecommendation
  - songs
    - __init__.py
    - featureSelection.py
    - model.py
    - myexception.py
    - test.py
    - test_data_cleaned.csv
    - test_data_for_song.csv
    - test_feature_selected_data.csv
    - Unit Test Result
      - Test Results - Nosetests_in_test.html
  - SongRecommendation.py
  - featureSelectedData.csv