Warsaw University of Technology

Project documenation

Music recommendation system

EINIS 2016L

Author: Michał Godek

Supervisor: prof. dr hab. inż. Mieczysław Muraszkiewicz

June 28, 2016

Contents

1 Task summary

The subject of the project is to create a recommendation system which will aid finding music tracks of ones liking. To achieve this, Collaborative Filtering together with Matrix Factorization technique will be used. Such an approach makes the solution domain agnostic and does not require to know feature details of ranked items. This makes it highly reusable for other content if only user preferences are available.

1.1 Goals and objectives

Main goals of the project are to:

- create a music recommender program, which outputs tracks based on an input playlist;
- evaluate the resulting output.

2 Task analysis

This section lists the elements constituting the solution of the problem at hand.

2.1 Data set

The essence part of the project is the data set on which the prediction model will be created and evaluated. The archive named 'The Million Song Dataset'[2] provides a listing of a million tracks along with their features and a real user preference set. In this project the features of the tracks are ignored, instead the focus is put on the users' taste in music. The data set used for the project consists of 2M user-track-rating triplets accompanied by a track listing of 1M items including EchoNest trackIds and track names.

2.2 Prediction model

Collaborative Filtering is an approach to finding new content for recommendation to a user, which depends mostly on 'the wisdom of the crowd', where suggested music comes from other users' playlists[4]. This project will utilize the idea presented in the paper "Matrix Factorization Techniques for Recommender Systems"[3] written by the winners of the 'Netflix Prize'. The concept behind it is that given a user - track preference matrix see table 2.1.

	Track 1	Track 2	Track 3	Track 4
User 1	7	?	1	?
User 2	?	?	?	4
User 3	9	0	?	2
User 4	1	8	?	0

Table 2.1: User - Track preference matrix

the algorithm searches for latent feature matrices see 2.2 on the next page, which when multiplied would create the input preference matrix. The latent matrices are calculate by iteratively improving them to find their optimal values and predict the missing values in the original user - item matrix.

	Feature 1	Feature 2
User 1	?	?
User 2	?	?
User 3	?	?
User 4	?	?

Table 2.2: Users - Latent features matrix

	Track 1	Track 2	Track 3	Track 4
Feature 1	?	?	?	?
Feature 2	?	?	?	?

Table 2.3: Latent features - tracks matrix

2.3 Evaluation method

Assessment of the results of a recommender system is not straight forward, since for true grading of the solution, real users would need to evaluate it and share their opinion on the usefulness of the program. In this project, the evaluation is limited to verifying the output playlist meets the subjective taste of the author.

2.4 Assumptions and constraints

User input playlist is given with a play count for each of the tracks. Play count is treated as a measure of liking a user has to each track. Due to limited data and computing resources, the input playlist must be generated by the user and based on the train data set. This limiation is posed due to the fact that it is hard to match enough tracks from a random user's playlist with tracks from the train set, even though the train set is very large.

3 Solution overview

3.1 Tools

The software was developed and tested to run on a Ubuntu Linux distribution equipped with Python 2.7.10. The whole program was implemented in Python and it was only accompanied by a Bash installation script. Thus, by only modifying the installation script, it may be easily adapted for other platforms. For assistance at accessing the Spotify Web API, the Spotipy library was used.

3.2 Program steps

- User with a Spotify account is requested to permit access rights for use by the music_recommendation_system. Access is required to gather user's listening data and to allow to present the program output as a new playlist.
- A playlist prepared by the user and based on the song_datacsv file is translated into Spotify tracks and is added to Spotify as a playlist.
- User's playlist is fetched from Spotify and is then used to filter the training data.

- Collaborative Filtering algorithm is applied to generate recommended tracks.
- An output playlist is generated and added to user's Spotify account.

4 Results evaluation

As mentioned before, it may be not objective to assess the results by user taste, but despite that, such an attempt was made. The generated playlist was found to be present fine recommendations. Most importantly, no outliers were present in the results, i.e. for rock tracks as input, no disco tracks were suggested. An important point to mention is the performance of the algorithm, which unfortunately is very time and resource consuming. During a typical test the data was limited to:

- 50 tracks in user's input playlist
- 500 unique users from the train set
- 1600 unique tracks from the train set

The time required to perform operation on data of such size was up to 30 minutes.

For on sight inspection given are the following links available after logging into Spotify as any user:

- input playlist: https://play.spotify.com/user/11122306349/playlist/7AKkMpIegJRatwzBRdpnaN
- output playlist: https://play.spotify.com/user/11122306349/playlist/4Cw4CPmPxYCRPHsEYcyK4a

4.1 Suggested improvements

- A few track features might be incorporated into the Collaborative Filtering model to aid the latent feature matrices creation.
- A possible extension is to use a much larger music data set and improve the computation time using either:
  - the Apache Spark library and deploying the solution to a cloud cluster;
  - the TensorFlow[1] library and utilizing the GPU of a local machine or deploying the program to a cloud for computing.

5 Program run instructions

Running the program is a three step process:

1. Run the setup.sh script to download data files and install dependencies
2. Manually prepare a file with user preference tracks. On Linux the following command allows to create such a file:

```
less song_data.csv | grep -i 'the police\|gilmour\|pink floyd\|depeche mo
de\|talking head\|peter gabrie' >> user_favorites.txt
```

3. Finally launch the program with command:

```
python main.py
```

6 References

[1] Katherine Bailey. Matrix factorization with tensorflow, 2016.

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[4] Albert Au Yeung. Matrix factorization: A simple tutorial and implementation in python, 2010.