SONG RECOMMENDATION SYSTEM

M Aishwarya
1BY17CS085 CSE VI B

Song Recommendation System

Python Mini Project Report

Introduction

With the growth of Artificial Intelligence and machine learning, we encounter algorithms every day from the moment we pick up our phone to search to when we decide what we want to buy, eat, where to go, etc. One of the most common algorithms used in popular sites like amazon, Netflix, YouTube, etc is there recommendation algorithm. What is recommendation algorithm? It is basically a suggestion. Generally, when we go to restaurant or shopping complex, we encounter salespersons, who cater to our needs from the time we enter the store to when we leave it. They ask our requirements and get relevant products from the store. In some cases, they may even recommend their store products that we may like. Now imagine the shop you visit is online, now who will greet you and fetch you the required products? It is common to observe chatbots, that can be with real people, or AI powered bots, that guide us through the site, but they are limited to time and availability in case of humans, and power of understanding natural language in case of bots. So, we need a solution that is independent of these limitations. This is where recommendation systems come in.

Concepts Involved

Recommendation algorithms are a group of algorithms that focus on grouping items based on some similarity factor. And then use these groupings to fetch related/ relevant items. The kind of similarity heuristic used determines the kind of recommendation algorithm. Mainly there are 2 kinds of recommendation systems,

- Groups based on items
- Groups based on users

Now we will look into these 2 aspects in more detail.

Grouping based on items : Now any item of interest be it movie, appliances, clothes, books, music, etc have some distinct features. Consider thee example of music. We can classify a song based on its genera e.g.: melody, hip hop, rap, rock, etc, based on the movie/band that played it, year of release, etc. Now we can group the songs together in clusters based on similarity of features. So, when a user likes a song from one cluster, we can assume, she will like other songs in the same cluster and recommend accordingly. For clustering these songs, we can use any of the many available clustering algorithms, the most popular choice being K means clustering algorithm.

The limitations of this approach to recommendation is that, we can recommend only those songs in that particular cluster, that is we cannot diversify. This will negatively impact user engagement with the platform, and may eventually lead to drop in user interaction with the platform over time.

Grouping based on users: Another approach to recommending items is to look at the items bought by one user, and then compare it to other users who bought the same and additional items, and then recommend those additional items to the first user. This method takes advantage of the knowledge gained when a user purchases item. It also is able to recommend products from varying categories, which will help in diversifying markets and improve user interaction. The main data requirement for this system is the rating given to products by users. This will help group users based on similar ratings. This kind of recommendation system is visible in Amazon, where once we add an item to cart it shows the message: 'Users that purchased this, also purchased…'. Nowadays, this is also visible in youtube's recommendation system where, after you watch/subscribe to a video it shows, viewers of this video also subscribe to x channel.

This seems like a good approach, but the limitation lies in the data. That is, we need a large number of user interactions, to be able to make valid predictions.

Thus, when implementing a recommendation system, we can start with content-based recommendation, and once enough data is collected, we can move to user-based recommendation.

So, this approach is suitable for an existing user, with previous purchases/interactions. But what if we have a new user, about whom we have no data. How can we recommend items to them? This is an area of ongoing research, and some ways to tackle this issue is:

- Recommend the most popular items, for example in music, recommend the latest top song, or in products, the most purchased product recently
- Ask the user to rate few items before hand and use that information to recommend, for example in music, play few songs and ask for rating, to build initial database.

We have explored the 2 basic ways in which recommendation systems work. Although in practice, more complex systems are used, which considers many more aspects. Also, more generally a combination of systems is used. There are many competitions and prizes for people/teams to develop high accuracy recommendation systems using machine learning and artificial intelligence. The most famous one, being Netflix's million-dollar prize challenge. Surprisingly the winning team's algorithm was not implemented by Netflix, as it did not scale well.

This draws to light another important aspect, we must not focus only on improving accuracy, as it may lead to unnecessarily complex models that do not scale well, that is the program can take a long time to run, or lead to server crash, when thousands or lakhs of users engage with the system. And a solution need not be complex to be right, even simple solutions can serve the purpose.

Inspired by the growing importance of recommendation systems, and the role they play everyday in our lives, I wanted to implement these concepts I learnt using python, to build a music recommendation system.

Describing the dataset

The dataset for music recommendation system was obtained from Kaggle. It contains 3 text files, one for list of songs, one for list of users, and one containing the rating given to certain songs by certain users

Code description

Step 1: First we create 3 dictionaries, to store the number of times each song was listened to, the list of songs heard by each user and the number of songs heard in a loop.

Step 2: Sort the songs based on frequency. To reduce the file size, we map the song name to integers

Step 3: We recommend 10 songs to a user, so we create a list to store the them. Then for song that is not in the list of songs already heard by the user, we add that song to the recommended song.

Step 4: Now we make some assumptions to calculate the similarity among users.

Assumptions: song being listened to more than once = favourite song

- group users based on similar song listened
- Find candidate set of songs
- order by loop factor

We use Jaccard similarity, to calculate the similarity among users. Jaccard similarity for 2 users is defined as the ratio of the intersection and union of songs listened to by the two users.

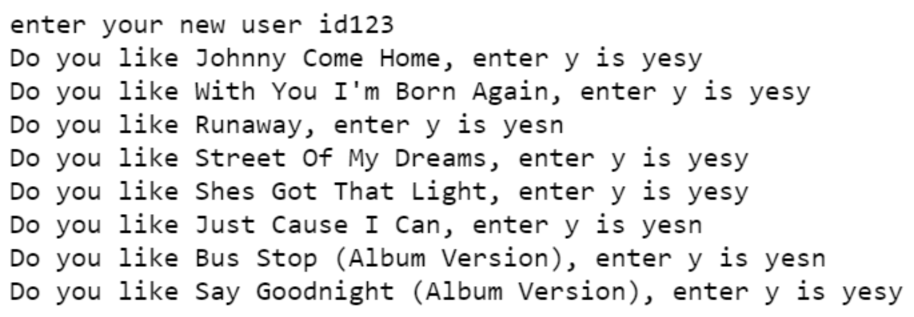
Step 5:

- We calculate the similarity for the user with the other users and sort the values. These similarity values are stored in a dictionary
- Next we find candidate songs to recommend by looking at songs heard by similar users.
- Then we sort the candidate songs by loop factor, to get the top recommendations

Step 6: The songs in the text file are stored as id numbers. We need to get song names, to be able to recommend to the user. To accomplish this, we use SQLite package in python to query the title, from the id.

Step 7: Now we can begin recommending the songs to user.

- First we ask the user to enter their user id. Then we perform steps 3,4 and 5 to get the top 10 recommended songs.
- Suppose the user is a new user, without user id then a new user id is created. We then ask the new user to rate 10 randomly selected songs. These ratings are stores and used to calculate similarity as in the previous case.



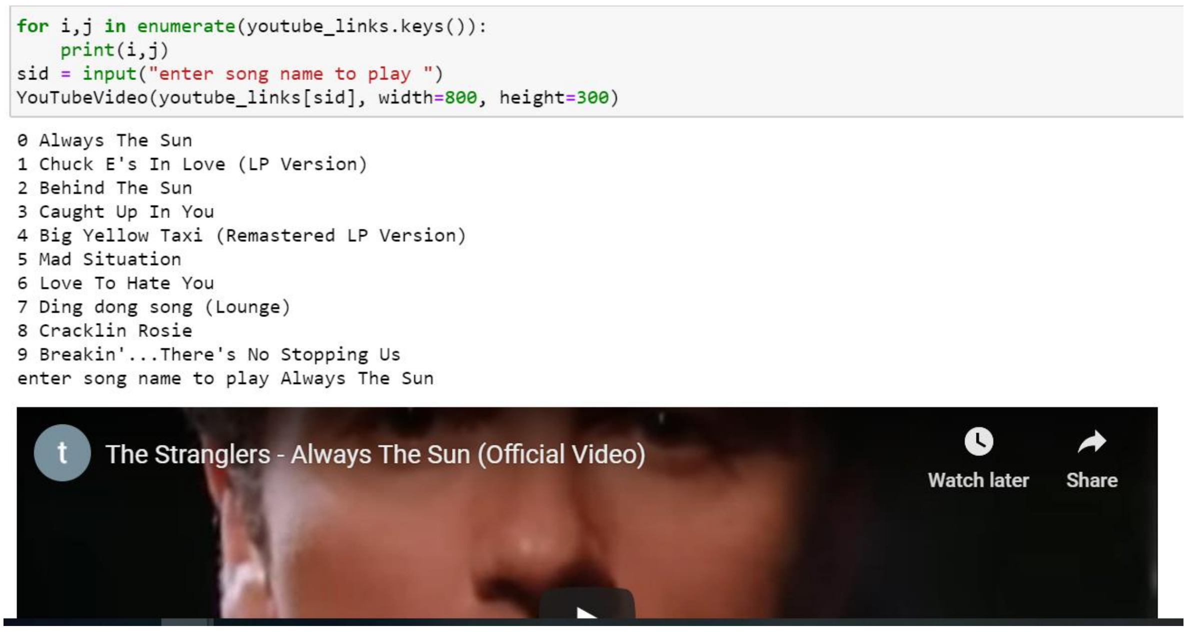
```
enter your new user id123
Do you like Johnny Come Home, enter y is yesy
Do you like With You I'm Born Again, enter y is yesy
Do you like Runaway, enter y is yesn
Do you like Street Of My Dreams, enter y is yesy
Do you like Shes Got That Light, enter y is yesy
Do you like Just Cause I Can, enter y is yesn
Do you like Bus Stop (Album Version), enter y is yesn
Do you like Say Goodnight (Album Version), enter y is yesy
```

Figure 1: Asking new user to rate few random songs

Step 8: Now that the recommended songs are obtained, we need to allow the user to listen to the song. To achieve this I used Beautiful Soup, json, requests and urllib packages in python to search for the recommended song on YouTube, and retrieve the relevant URL.

Step 9: The above step is accomplished by creating a class called YouTube Search, that has 3 attributes, search terms, maximum number of results (i.e. 10) and video. It has methods to search, parse HTML and get the video.

Step 10: The retrieved music video for the top recommendation is played for the user to enjoy.



```
for i,j in enumerate(youtube_links.keys()):
    print(i,j)
sid = input("enter song name to play ")
YouTubeVideo(youtube_links[sid], width=800, height=300)
```

```
0 Always The Sun
1 Chuck E's In Love (LP Version)
2 Behind The Sun
3 Caught Up In You
4 Big Yellow Taxi (Remastered LP Version)
5 Mad Situation
6 Love To Hate You
7 Ding dong song (Lounge)
8 Cracklin Rosie
9 Breakin'...There's No Stopping Us
enter song name to play Always The Sun
```

Figure 2: Screenshot of recommended songs for new user (List of top 10 recommended songs, along with video)

Hence, using the concepts learnt about recommendation systems, along with the programming concepts learnt in python, I have implemented an interactive music recommendation system, that recommends songs based on similarity among users.