MUSIC RECOMMENDATION SYSTEM

HAVING CONTENT-BASED AND COLLABORATIVE FILTERING

PROJECT REPORT for MACHINE LEARNING (IT-323)

Minor-Project

By

SARTHAK JAIN 2K19/ME/216

HARSHAVARDHAN SINGH 2K19/ME/103

Under The Guidance of

Prof. Dr. Dinesh K Vishwakarma

Department of Information Technology

Delhi Technological University

2021

ACKNOWLEDGEMENT

This project has in due course helped us to learn many concepts which surround the rapidly emerging fields of Machine learning and deep learning. We feel grateful to do this as our Machine Learning (IT-323) minor-project and Prof.Dr. Dinesh K Vishwakarma as our project guide. We are thankful to him for his help and guidance. He has always been kind to aid us whenever we got stuck. Finally we would like to thank our family and friends who stood by us all the time and cooperated in the best way they could. There is a great sense of satisfaction in completing this project.

SARTHAK JAIN

HARSHVARDHAN SINGH

Machine Learning- An Introduction

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition**,** speech recognition**,** email filtering**,** Facebook auto-tagging**,** recommender system**,** and many more.

The term Machine Learning was coined by Arthur Samuel in 1959, an American pioneer in the field of computer gaming and artificial intelligence, and stated that "it gives computers the ability to learn without being explicitly programmed".

A Machine Learning algorithm is a set of rules and statistical techniques used to learn patterns from data and draw significant information from it. It is the logic behind a Machine Learning model. An example of a Machine Learning algorithm is the Linear Regression algorithm.

Music Recommendation Algorithm: Beginning

Through this project we have tried to design an algorithm that could recommend music to its users by analyzing a given set of database.

We have **used content based and collaborative filtering system** to write the algorithm for streamlining the process of music selection and provide the desired results.

The content based system (CBRS) recommends items based on its feature and the similarity between elements of other items. Assume a user has already listened to a song by a particular artist or a particular genre say "rock" then the CBRS will recommend more songs from the same genre or artist.

```
#Get similar items to given items
def get_similar_items(self, item_list):

    user_songs = item_list

    #B. Get all unique items (songs) in the training data
    all_songs = self.get_all_items_train_data()

    print("no. of unique songs in the training set: %d" % len(all_songs))

    #C. Construct item cooccurence matrix of size
    #len(user_songs) X len(songs)
    cooccurence_matrix = self.construct_cooccurence_matrix(user_songs, all_songs)

    #D. Use the cooccurence matrix to make recommendations
    user = ""
    df_recommendations = self.generate_top_recommendations(user, cooccurence_matrix, all_songs, user_songs)

    return df_recommendations
```

The Content Based Recommendation system used

To address some of the limitations of content-based filtering, collaborative filtering uses *similarities between users and items simultaneously* to provide recommendations. This allows for serendipitous recommendations; that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user B.

The collaborative filtering system (CFRS) recommends music on the basis of users choice or the pre selections that the user has made. Suppose the user selects that they like jazz music then the algorithm will make sure the jazz genre is recommended to the user

```
def construct_cooccurence_matrix(self, user_songs, all_songs):

    #Get users for all songs in user_songs.
    user_songs_users = []
    for i in range(0, len(user_songs)):
        user_songs_users.append(self.get_item_users(user_songs[i]))

    #Initialize the item cooccurence matrix of size
    #len(user_songs) X len(songs)
    cooccurence_matrix = np.matrix(np.zeros(shape=(len(user_songs), len(all_songs))), float)

    #Calculate similarity between user songs and all unique songs
    #in the training data
    for i in range(0,len(all_songs)):
        #Calculate unique listeners (users) of song (item) i
        songs_i_data = self.train_data[self.train_data[self.item_id] == all_songs[i]]
        users_i = set(songs_i_data[self.user_id].unique())

        for j in range(0,len(user_songs)):

            #Get unique listeners (users) of song (item) j
            users_j = user_songs_users[j]

            #Calculate intersection of listeners of songs i and j
            users_intersection = users_i.intersection(users_j)

            #Calculate cooccurence_matrix[i,j] as Jaccard Index
            if len(users_intersection) != 0:
                #Calculate union of listeners of songs i and j
                users_union = users_i.union(users_j)
                cooccurence_matrix[j,i] = float(len(users_intersection))/float(len(users_union))
            else:
                cooccurence_matrix[j,i] = 0


    return cooccurence_matrix
```

IMPLEMENTATION OF THE ALGORITHM

Machine Learning algorithms learn from data. They find relationships, develop understanding, make decisions, and evaluate their confidence from the training data they're given. And the better the training data is, the better the model performs.

The dataset used to train the model is the "million song dataset which has around 2 million observations. we used a part of it to train our algorithm for recommending music

the dataset has been imported to a pandas dataframe. Then using functions available through the pandas library, the dataframe has been grouped by the parameter 'listen_count'.

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the **data**, **rows**, and **columns**. Pandas DataFrame will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, and Excel file.

```
1 # Calculating the listen count by song
2 grpbysong = song_data.groupby('title')[['listen_count']].sum()
3 print(grpbysong)
```

```
                                                listen_count
title
#!*@ You Tonight [Featuring R. Kelly] (Explicit...          148
#40                                                        1831
& Down                                                     739
' Cello Song                                               254
'97 Bonnie & Clyde                                         140
...                                                        ...
the Love Song                                              279
you were there with me                                      89
¡Viva La Gloria! (Album Version)                           538
¿Lo Ves? [Piano Y Voz]                                     191
Época                                                      443

[9567 rows x 1 columns]
```

Pandas Dataframe used in the algorithm

The dataframe obtained after the grouping can be sorted in the descending order to have insights on listening pattern of users and to build a 'popular playlist'.

The song_data dataframe is then used to buid a cooccurence matrix, using the result of which the model recommends songs to the user.

The Co-Occurrence Matrix

A correlation matrix is a table containing correlation coefficients between variables. Each cell in the table represents the correlation between two variables.

The value lies between -1 and 1. The function construct_cooccurence_matrix() is passed a list of all the songs listened by the user (user_songs). Then we make a list of the users, who have also listened to those songs. it then calculates the similarity between the songs which the user has listened to in the past and all other unique songs(all_songs).

It does so by taking a unique song and then finding the intersection of users, who listen to the unique song and one of the (user_songs). For every iteration, it calculates the value of similarity to be store in the cooccurence matrix using Jaccard Index.

The Jaccard Index, also known as the Jaccard similarity coefficient, is a statistic used in understanding the similarities between sample sets. The measurement emphasizes similarity between finite sample sets, and is formally defined as the size of the intersection divided by the size of the union of the sample sets. The mathematical representation of the index is written as:

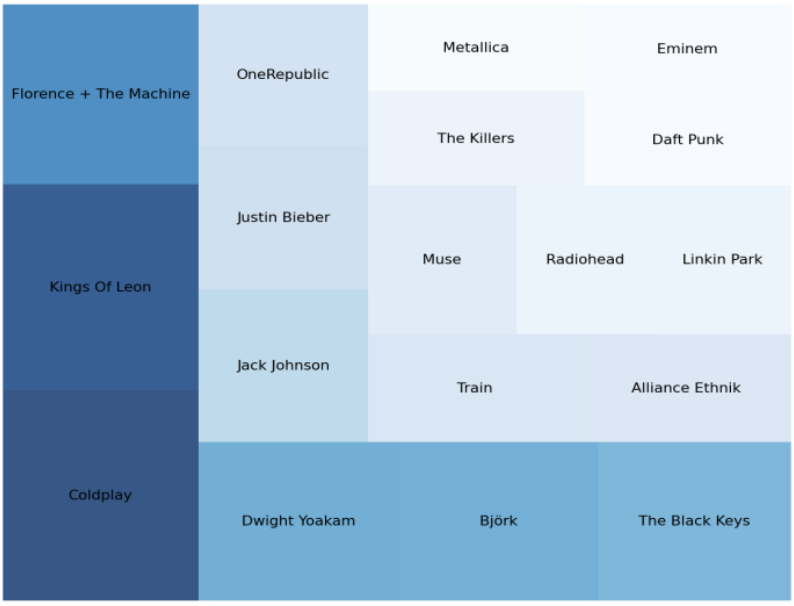
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

LINK TO THE GITHUB REPOSITORY

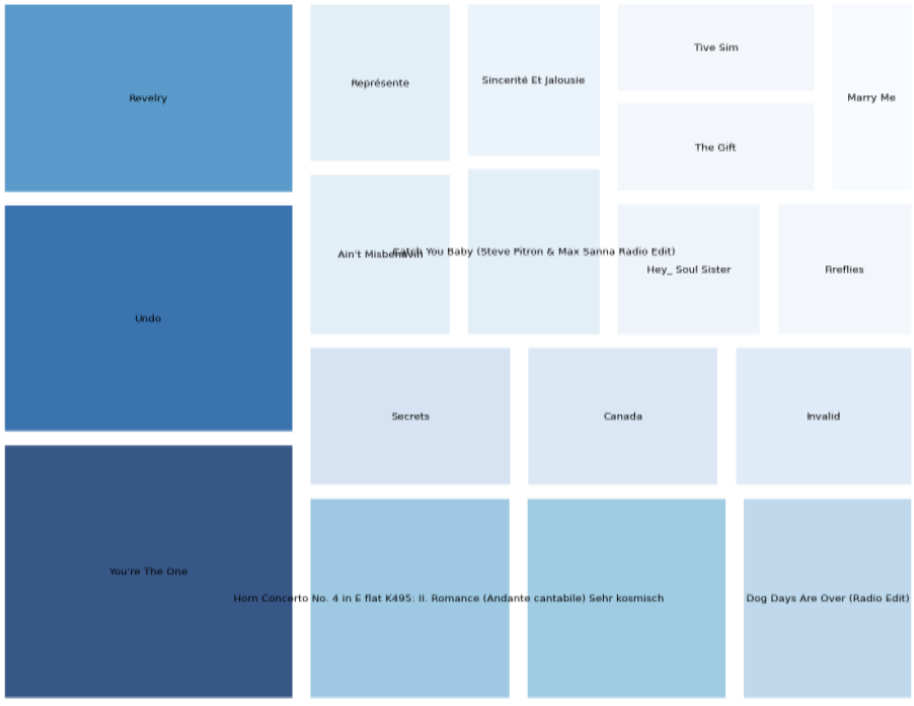
https://github.com/ShadyPsychonaut/music-recommendation-system

RESULTS AND CONCLUSION

Using methods provided by the pandas library, we were able to curate a 'Popular Playlist' and the model was able to identify songs based on the listening preferences of the user and was able to recommend other songs based on the information.



Popularity Result based on Artist_name



Popularity Result based on title

The model was able to identify the similarity coefficient between two songs and hence by comparing the user's choice similarity with others in the database it was able to pick appropriate songs to suggest the user

Content based and collaborative filtering systems worked as expected and gave the desired results.

To conclude we can say that algorithms could be successfully trained to recommend the right music to a user making their listening experience better and have certain positive associated impacts

The right type of music can affect a person in more than one positive ways increasing his or her productivity by relaxing the brain and through our algorithm we have inched a step closer to create a better music recommendation algorithm.

Result based on User-based collaborative filtering-

```
# recommend similar songs to the user based on his/her listening history
mcr.recommend(song_data['user_id'][40])
```

```
No. of unique songs for the user: 45
no. of unique songs in the training set: 9953
Non zero values in cooccurence_matrix :268460
```

	user_id	song	score	rank
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Quiet Houses - Fleet Foxes	0.034172	1
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Meadowlarks - Fleet Foxes	0.033473	2
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Heard Them Stirring - Fleet Foxes	0.032683	3
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Great Indoors - John Mayer	0.032123	4
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Tiger Mountain Peasant Song - Fleet Foxes	0.031740	5
5	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Sun It Rises - Fleet Foxes	0.031253	6
6	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Your Protector - Fleet Foxes	0.030409	7
7	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Oliver James - Fleet Foxes	0.030237	8
8	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Belle - Jack Johnson	0.028708	9
9	b80344d063b5ccb3212f76538f3d9e43d87dca9e	If I Could - Jack Johnson	0.028350	10

Result based on Song-Based Collaborative Filtering-

```
# finds similar songs based on the entered song
mcr.get_similar_items(['Stacked Actors - Foo Fighters'])
```

```
no. of unique songs in the training set: 9953
Non zero values in cooccurence_matrix :5078
```

	user_id	song	score	rank
0		Generator - Foo Fighters	0.299539	1
1		Breakout - Foo Fighters	0.210863	2
2		Next Year - Foo Fighters	0.209016	3
3		Weenie Beenie - Foo Fighters	0.160550	4
4		X-Static - Foo Fighters	0.157143	5
5		For All The Cows - Foo Fighters	0.150628	6
6		Floaty - Foo Fighters	0.148760	7
7		No Way Back - Foo Fighters	0.148325	8
8		Oh_ George - Foo Fighters	0.146789	9
9		Low - Foo Fighters	0.142857	10

REFERENCES:

http://millionsongdataset.com/pages/getting-dataset/#subset

https://www.geeksforgeeks.org/hdf5-files-in-python/

https://www.askpython.com/python/examples/plot-a-treemap-in-python

https://cmdlinetips.com/2019/03/how-to-select-top-n-rows-with-the-largest-values-in-a-columns-in-pandas/

https://thispointer.com/pandas-convert-dataframe-index-into-column-using-dataframe-reset_index-in-python/

J. Ben Schafer, Dan Frankowski, Jon Herlocker , and ShiladSen(2007) “Collaborative Filtering Recommender Systems”

https://www.geeksforgeeks.org/plotting-correlation-matrix-using-python/

https://pandas.pydata.org/