Specialist Diploma in Applied Artificial Intelligence - Nanyang Polytechnic   
26 Aug 2020

**Final Report**

[**Song Recommendation System for Machine Learning using Spotify** ]

**Project Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team 9** | Liang Chee Wei, Kenneth (20A453C) | Rohit Kumar (20A465U) | Padmapriya Mathivanan (19B683N) |
| **Project Mentors** | Group A  Mr. Mar Kheng Kok,Mr. Lee Chit Boon, Mr. Lee Ching Yuh, Mr. Law Chee Yong | | |

1. **Introduction**

Music recommendation and playlists continuation are major challenges that music streaming services such as Spotify face when trying to meet rising customer demands for a good online experience. Spotify estimates that 60% of the time, listeners know what they want to listen and they just need to find it [5]. The remaining 40% of the time, listeners are receptive to new ideas but in a passive and impatient state. Such listeners are described as being in an **“open” mindset**.

It is therefore important for the lasting appeal of music streaming services to build a recommender system that caters to this behaviour to enhance user experience.

1. **Formulation and Method**

In this project, the team will attempt to develop a recommender system for automating playlist generation by the addition of relevant songs, hence extending a current playlist beyond the last song. Our focus is on enhancing user experience when listeners are in an ‘open’ mindset (40% of the time), as stated above.

The recommender system will be using machine learning methods based on the similarity (specifically, cosine similarity) between recommended tracks and the ‘seeded’ (current)tracks from starting playlists. The recommender system will use machine learning approaches such as content filtering and collaborative filtering to predict which songs are the “best match” to the songs in the starting playlists. The ideal outcome is to suggest a list of 10 songs to add to the playlist, that is most relevant to user after user has listened to songs on a playlist.

Given that the ground truth is the complete set of tracks of each playlist from the original dataset, and that prediction is done on the subset of playlist (after subtracting some tracks for validation and test set), the machine learning model will have demonstrated success if:

At least one track (out of 10 predicted) can be matched to the tracks in the set-aside playlist in the test set. The assumption is that user can only listen one song at a time, therefore, the next recommended song matters more than a list in the extreme minimal case.

1. **Intended Experiments**

3 models will be explored, starting with a baseline model. These will be explained below.

The **baseline model** will use two/three predictors from Spotify API’s audio features (There are 20 audio features altogether for each track). An experiment will be required to choose the more suitable audio features. Using the simplified set of information, a baseline model will utilise SVM or K-Nearest Neighbours algorithm (to be decided) to suggest songs with the closest feature scores to the average of the existing songs in the playlist.

Two further models using **content filtering and collaborative filtering approaches** will be develop and compared against the baseline model. Content filtering approach will utilise the full 20 audio features for each track in playlists.

Collaborative approache will require reinterpretation of user-item collaborative filtering approach. In traditional

perspective of user-item collaborative filtering, the model will look at users who have similar behaviors to another particular user based on similar ratings and recommend items/product those similar users also liked. Essentially, “Users who are similar to you, also liked ….” approach.

For our problem, we will treat “user” as “playlist” and “item” as “track”(i.e. song). This approach was taken based on data provided from the selected dataset. We will also reinforced the selected dataset with further data extracted from Spotify API.

The assumption of this approach for collaborative filtering model is based on idea that we should select songs from playlists which are similar to the starting playlist. The second part of the assumption is that similar songs from playlists which are similar to starting playlists are more similar to songs in starting playlists.

**Evaluation Of Machine Learning Algorithm**

One of the key challenges in development of recommendation system is evaluating its impact. It is possible to generate tracks to recommend to user, but there is a need to quantify its impact, hence its relevance to the user. In the review of existing literature [6] & [7], two measure stand out as more relevant. The following two evaluation metrics will be use to compare the 3 models (baseline, content filtering, collaborative filtering).

1. Normalized Discounted Cumulative Gain(NDCG)

NDCG measures the usefulness or gain of a ‘track’ based on its position in the playlist. The assumption is that highly relevant ‘tracks’ are more useful if they appear earlier in the playlist. And highly relevant ‘tracks’ are more useful than marginally relevant ‘tracks’.

1. R-Precision

R-precision is the number of recommended relevant tracks divided by the number of known relevant tracks (i.e. the number of tracks set aside in the validation or test set). (R represents the recommended ordered list; G represents the true user-created playlist; The size of a list is denoted by | |.)

A screenshot of a cell phone

Description automatically generated

1. **Relevant Dataset and Prior Research**

There are various similar projects work published and discussed over the internet whereby people had shared the process of creating a recommendation system through Spotify API

**Data Collection:** We will be using the dataset from [here](https://github.com/vaslnk/Spotify-Song-Recommendation-ML/tree/master/data) **(https://github.com/vaslnk/Spotify-Song-Recommendation-ML/tree/master/data)**

About the Dataset:

The dataset is from Spotify Million Playlists (Recsys 2018) Challenge [2]. The dataset contains a million user-generated playlists. These playlists were created during the period of January 2010 through October 2017. Each playlist in the MPD contains a playlist title, the track list (including track metadata) editing information (last edit time, number of playlist edits) and other miscellaneous information about the playlist.

In addition, we will be using Spotify API to extract 20 audio features for each track.

**References From Online Resources Prior Research**

1. Spotify Music Discovery

<https://github.com/adidottxt/spotify-music-discovery>

1. Spotify Million Playlists (RecSys 2018) Challenge Submission

<https://github.com/vaslnk/Spotify-Song-Recommendation-ML>

1. Get Recommendations Based on Seeds – Spotify for developers <https://developer.spotify.com/documentation/web-api/reference/browse/get-recommendations/>
2. Recsys challenge 2018: automatic music playlist continuation <https://www.researchgate.net/publication/327949464_Recsys_challenge_2018_automatic_music_playlist_continuation>
3. How Spotify Recommends Your New Favorite Artist. <https://towardsdatascience.com/how-spotify-recommends-your-new-favorite-artist-8c1850512af0>
4. Ben Lai, Ziyuan Zhong, Qianqian Hu. Columba University:

Music Playlist Continuation Via Deep Variabtional Embedding Augmented Neighbour Search.

<https://drive.google.com/file/d/1wmNnkb9rOetCNGp4m5WbT_I0Fh7tANld/view>

1. Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys challenge 2018: Automatic music playlist continuation. InProceedings of the 12th ACM Conference on Recommender Systems, pages 527–528.ACM, 2018.

<https://www.jku.at/fileadmin/gruppen/173/Research/chen_recsys_2018.pdf>

**Experiment 1: Using SVM Classifier** **to recommend song to a playlist**

**Individual Contribution**

|  |  |
| --- | --- |
| Name | Model |
| Rohit Kumar (20A465U) | Will be Using SVM Classifier to recommend song to a playlist |

**1. Data Analysis and exploration:**

In the Data Analysis and exploration part we will be exploring the characteristics of our dataset. The objective is to pick the most suited features that are going to be crucial in the learning of our Model.

1. The initial step would be to check for consistency of dataset to make sure there are no empty and null values
2. The next step would be finding out if there are any duplicate playlists in our dataset if found we found remove them
3. After the step (b) we would explore the song counts within a playlist throughout our dataset
4. Upon exploration we will find the statistical figure about how many songs are within each playlist
5. For now, we plan to pick any 20 playlists with songs more than 150 each. The playlist must be with similar song counts to maintain balance in our classes. [ We plan to explore further how we can refine the method of choosing 20 playlist ]

**2. Data Pre-Processing**

We will start by dropping all the features which are not relevant to our model.

**Features to be removed**

pid', 'collaborative', 'modified\_at', 'num\_tracks', 'num\_albums','num\_followers', 'num\_edits','duration\_ms', 'num\_tracks.1', 'pos', 'track\_uri', 'artist\_uri', 'track\_name', 'album\_uri', 'duration\_ms.1', 'album\_name'

After removal of the above features we will left with the below features where playlist\_name is our target

**A screenshot of a cell phone

Description automatically generated**

**3. Machine Learning**

By now we have our dataset ready and also have gained some useful insight from it. We will now go forward with our ML algorithm which is SVM. We will train our model and predict the song to a playlist which most likely is our recommendation of a song to the playlist.

We will be performing following steps for the final objective

1. We will start with building multiple pipelines to gain maximum accuracy for our Classifier.
2. We will using different Scaling techniques (MinMax, Standard) and Normalizer to find out which one provides the best performance.
3. After we will be performing cross-validation to obtain accuracy scores of our Classifier.
4. After obtaining the accuracy scores we will know for sure which Scaling has performed better than the others
5. We plan to improve our accuracy score by tuning its hyper-parameter, by performing a GridSearch to find best values for C and Gamma.

**4. Scope of Improvement:**

1. Further exploration is required to find most suited playlists from our dataset to train our Classifier.
2. We are still exploring possibilities of using other ML algorithms lilke KNN.

**5. References:**

1. <https://towardsdatascience.com/a-music-taste-analysis-using-spotify-api-and-python-e52d186db5fc>
2. <https://github.com/jmcabreira/A-Music-Taste-Analysis-Using-Spotify-API-and-Python>.

**Experiment 2: Content Based Filtering For Playlist Recommendation**

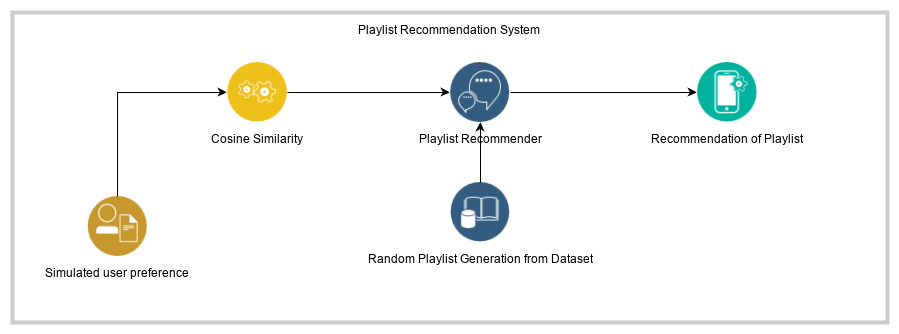
**Individual Contribution:**

|  |  |
| --- | --- |
| Name | Model |
| PADMAPRIYA MATHIVANAN (19B683N) | Content based filtering for playlist recommendation system |

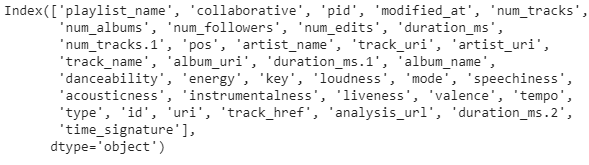
**Content Based Filtering for Playlist recommendation using Spotify dataset:**

Content-based method gives recommendations based on the similarity of two Playlist or attributes.

|  |  |
| --- | --- |
| * Data Analysis, Coding, Final report | <https://github.com/padmapriya-mathivanan/MachineLearning/tree/master/Recommender_System> |

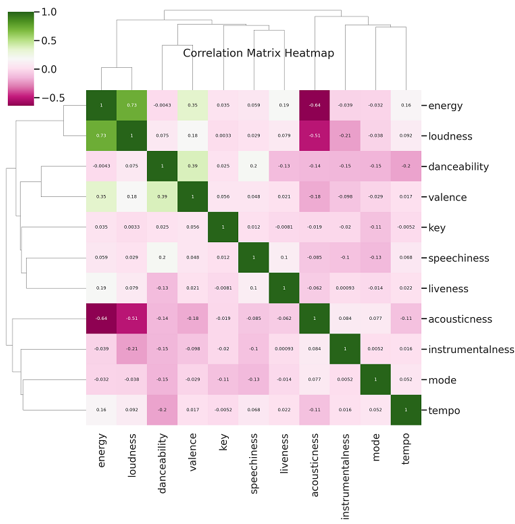


**Data Analysis: Attributes**



Number of unique Playlist – 476 – Key for content based filtering

**Correlation between track details:**



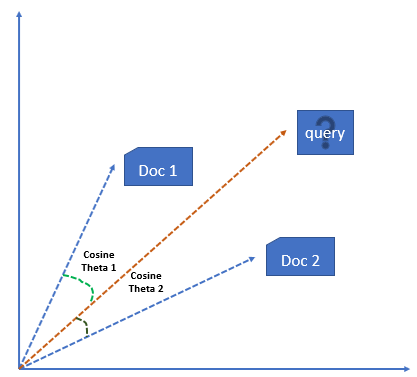
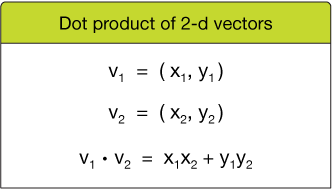
Functionality:

Recommender system has to decide for information delivery when providing the user with recommendations:

**Exploitation.** The system chooses documents similar to those for which the user has already expressed a preference.

**Vector Space Model**

In this model, each item is stored as a vector of its attributes (which are also vectors) in an n-dimensional space, and the angles between the vectors are calculated to determine the similarity between the vectors.

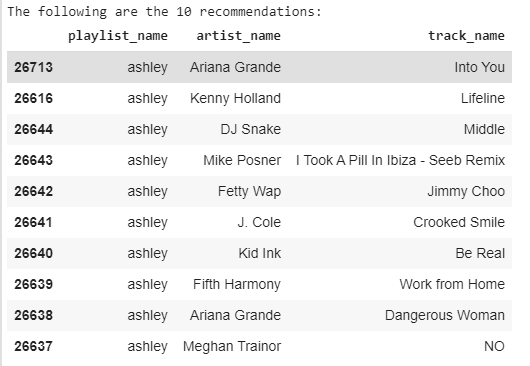
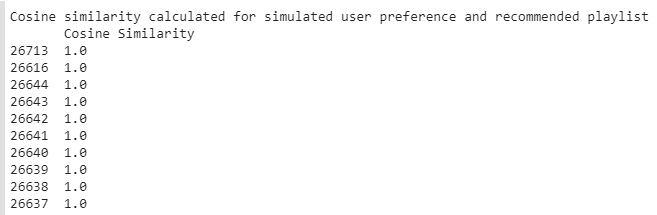
1. **Building the system for Content based filtering – Steps Involved**

* Content based filtering is a simple method used to recommend merely could be a string matching. In our dataset have identified the key to be the playlist using which the content based similarity is to be done.
* Building subset of playlist column with playlist binary values. [One-Hot-Encode](https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f) the list of playlist. We store every different playlist in columns that contain either 1 or 0.

[Number of Distinct playlist 476]

* Merge Initial Data frame with encoded Playlist Data frame
* Random shuffle is done to recommend random 5 playlist from the entire dataset
* One favourite playlist is chosen from this random list by user (in this approach we use simulated user preference) which is then used to get the cosine similarity and the list of 10 recommendations is done.

**Sample recommendations:**



1. **Pros and cons of content based filtering:**

**Pros:**

* The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
* The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

**Cons:**

* Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.
* The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

**References**:

<https://heartbeat.fritz.ai/recommender-systems-with-python-part-i-content-based-filtering-5df4940bd831>

<https://medium.com/@bindhubalu/content-based-recommender-system-4db1b3de03e7>

**Experiment 3: Collaborative Filtering For Playlist Recommendation**

**Individual Contribution:**

|  |  |
| --- | --- |
| Name | Model |
| LIANG CHEE WEI, KENNETH (20A453C) | Collaborative filtering for playlist recommendation system |
| Jupyter Notebook File | <https://drive.google.com/file/d/1fuZr7NPdHqJryoWiW6KwiSZ29Z8ciSd-/view?usp=sharing> |
| Jaccard Similarity Output File | https://drive.google.com/file/d/1fuZr7NPdHqJryoWiW6KwiSZ29Z8ciSd-/view?usp=sharing |

This section will detail the work done on collaborative filtering.

**1. General Approach**

Due to nature of available dataset which lack ratings score for user to songs, the approach will require reinterpretation of user-item collaborative filtering approach.

For our problem, we will treat “user” as “playlist” and “item” as “track”(i.e. song). This approach was taken based on data provided from the selected dataset. We will also reinforced the selected dataset with further data extracted from Spotify API.

The assumption of this approach for collaborative filtering model is based on idea that we should select songs from playlists which are similar to the starting playlist. The second part of the assumption is that similar songs from playlists which are similar to starting playlists are more similar to songs in starting playlists.

**2. Initial Data Exploration**

An initial extraction of the data from Spoitfy API gave a dataset of size 19010 songs with 36 features. A data exploration was conducted. There was no missing data in all feature columns.

Initial Dataset:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Number of Playlists | Number of Unique Playlists | Number of Unique Albums | Number of Unique Tracks | Number of Unique Artists | Number of Collaborative Playlist | Number of Duplicated Tracks | Average Playlist Length |
| 300 | 287 | 8343 | 13377 | 4587 | 9 | 5633 | 66.24 |

The initial approach to develop the train, validation and test set requires splitting the tracks accordingly. To ensure sufficient training instances, a judgement was made to select playlist with minimum of 100 songs. (ie. At least 60 songs for train, 20 for validation, 20 for test). During exploration of data, it was discovered that the number of playlist from initial dataset that fulfils this criteria is less than 50%, resulting in only about 150 suitable playlists. This is a low number.

As such, another round of data extraction was executed to increase the initial dataset size.

Final Dataset:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Number of Playlists | Number of Unique Playlists | Number of Unique Albums | Number of Unique Tracks | Number of Unique Artists | Number of Collaborative Playlist | Number of Duplicated Tracks | Average Playlist Length |
| 506 | 476 | 12379 | 20827 | 6574 | 13 | 12890 | 70.83 |

Other features were also explored to determine if it’s useful for collaborative filtering work. This include features such as ‘Number of Collaborative Playlist’, ‘Number of Edits’ and ‘Number of Followers’. ‘Number of Collaborative Playlist’ has too few data instances, only 13. ‘Number of Edits’ represent the number of times playlist is edited. Hence, it represent ‘active’ playlists. This may contain useful information. ‘Number of Followers’ represent the number of users who follow a particular playlist. Unfortunately , from data exploration, the overwhelming majority of playlist has only 1 follower. This represent that the dataset is extracted from individual users and are mostly not shared playlist among users.

**3. Playlist Based Collaborative Filtering**

The approach is to consider songs appearing in original playlist as a “vote” for the playlist. This is a similar idea to “Liked” or “Disliked” rating in typical user-item collaborative filtering. So the rows of the data matrix will be playlist and the columns will be songs. A song that appears in a playlist will be represented as a ‘1’ and given ‘0’ otherwise. The motivation for this approach is that we don’t have individual playlist ‘rating’ each song.

Since the dataset does not have this information, the Jaccard Similarity Index for binary data can be used, to do implicit similarity based on playlists commonality, in terms of playlists who have similar songs.

A critical consideration is that Jaccard coefficient measures similarity between finite sets. In order to obtain finite sets of information for each playlist, we have to transform existing dataset in two steps. First step is to one-hot-encode all 20827 unique songs. Second step is to handle duplicate playlist in the dataset. Duplicate playlists will add same songs more than once. So playlists are grouped and songs which appear more than once are only given a value of ‘1’, not more.

**4. Coping With Computation Intensity**

Because of the nature of computation of Jaccard Similarity index, it is quite computationally intensive to compute 1x20827 per playlist to 476 unique playlists.

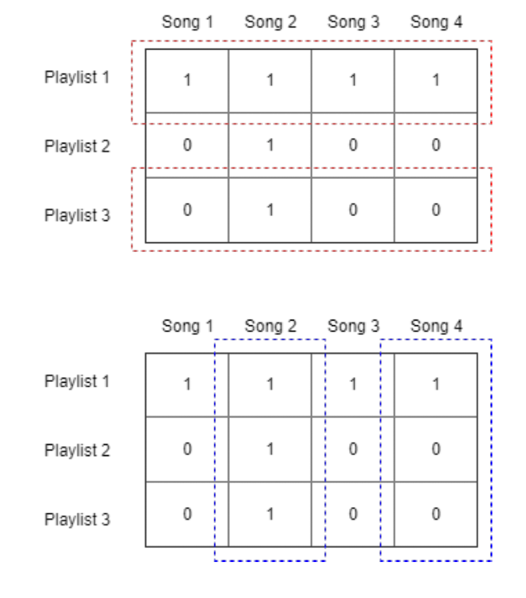
Thus, to handle the situation, the ‘multiprocessing’ package in Python is used. Python multiprocessing has two main classes, Pool and Process. The Pool class is utilized here. Pool distributes the task to available processors in FIFO scheduling . To decide on choice of Pool class, transformed data (in preparation to calculate Jaccard Similarity Index), and the nature of Jaccard calculation is considered. It’s evident that calculation of Jaccard index can be computed between playlists in parallel with computation of other playlist-playlist. There is no dependency between outcomes of computation or a data flow in an sequential effect. Hence, it is possible to create a Pool with number of processes as available CPU cores and pass the list to Pool.

A total of 7 CPU cores are utilized and the total computation time taken is about 1358 seconds, to compute Jaccard Index for the entire 476x20827 dataset.

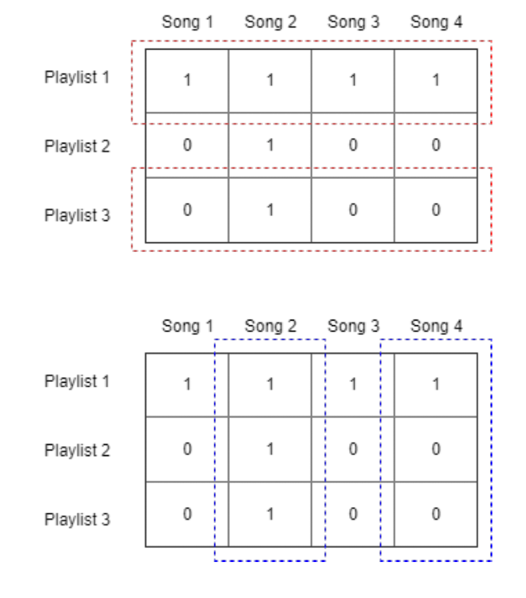
**5. Song Based Collaborative Filtering**

So far we have completed the basic step for playlist based collaborative filtering. The outcome from playlist based collaborative filtering is to obtain similar playlists. This enable a selected playlist to be able to identify with other similar playlists. But these similar playlists have other songs which are not on selected playlists. So there has be to a way to pick a song to add to selected playlist.

Song based collaborate filtering is required. This is analogous to item-based collaborative filtering. Essentially both Song based collaborate filtering and playlist based collaborative filtering uses the same data matrix, as illustrated:



Playlist based collaborative filtering



Song based collaborative filtering

**6. Evaluation Step**

The evaluation metric has been mentioned in previous section of this report and will not be repeated again here. Essentially the two evaluation metric used will be :

1. Normalized Discounted Cumulative Gain(NDCG).

<https://towardsdatascience.com/evaluate-your-recommendation-engine-using-ndcg-759a851452d1>

1. R-Precision