Automated Playlist Extension: Music Recommendation with Spotify Million Playlist Data

Zihan Ni
Cornell Tech zn43@cornell.edu

Abstract

Currently music service providers have generic (and popular), mood-based playlists, that are the same for all users. The goal of this paper is to introduce a music recommendation system that can provide customized music recommendation for any given playlist based on two classic approaches: collaborative filtering and word2vec embedding. By suggesting appropriate songs to add to a playlist, a Recommender System can increase user engagement by making playlist creation easier, as well as extending listening beyond the end of existing playlists. Here, we device a system which retrieves the similarity between playlists and recommend music based on the similarities. The link to github: https://github.com/zn8ae/RecSys2018

1 Introduction

Traditionally, Spotify has relied mostly on collaborative filtering approaches to power their recommendations. The idea of collaborative filtering is to determine the users preferences from historical usage data. For example, if two users listen to largely the same set of songs, their tastes are probably similar. Conversely, if two songs are listened to by the same group of users, they probably sound similar. This kind of information can be exploited to make recommendations.

Pure collaborative filtering approaches do not use any kind of information about the items that are being recommended, except for the consumption patterns associated with them: they are content-agnostic. This makes these approaches widely applicable: the same type of model can be used to recommend books, movies or music, for example.

In this work, we are also using Word2Vec, which is a technique commonly used in the Natural Language Processing domain. The idea behind Word2Vec is pretty simple. Were making an assumption that the meaning of a word can be inferred by the company it keeps. If you have two words that have very similar neighbors (meaning: the context in which its used is about the same), then these words are probably quite similar in meaning or are at least related. For example, the words shocked, appalled and astonished are usually used in a similar context.

The data we are given is the Million Playlist Dataset (MPD) from Spotify. Each playlist contains information such as name, description, number of songs, as well as the tracks information which entails the details about each track in this playlist. However, no user information is given, nor do we have any subjective attitude expressed towards these playlists such as ratings, listening behaviors, etc.

This paper is organized as follows: First, we will describe the dataset, how do we clean and organize the data, as well as the key insight retrieved from the data set. Then, we will describe the structure of our collaborative filtering model and Word2Vec model. Lastly, we will present our experimental setup along with some results generated by our model.

2 Dataset

2.1 Source

The MPD 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 in-

formation (last edit time, number of playlist edits) and other miscellaneous information about the playlist. Playlists that meet the following criteria are selected at random:

- Created by a user that resides in the United States and is at least 13 years old
- Was a public playlist at the time the MPD was generated
- Contains at least 5 tracks
- Contains no more than 250 tracks
- Contains at least 3 unique artists
- Contains at least 2 unique albums
- Has no local tracks (local tracks are non-Spotify tracks that a user has on their local device)
- Has at least one follower (not including the creator)
- Was created after January 1, 2010 and before December 1, 2017
- Does not have an offensive title
- Does not have an adult-oriented title if the playlist was created by a user under 18 years of age

2.2 Insights

As we conduct our explanatory data analysis, we have found the following aggregate statistics about the dataset:

- number of playlists 1000000
- number of tracks 66346428
- number of unique tracks 2262292
- number of unique albums 734684
- number of unique artists 295860
- number of unique titles 92944
- number of playlists with descriptions 18760
- number of unique normalized titles 17381
- avg playlist length 66.346428

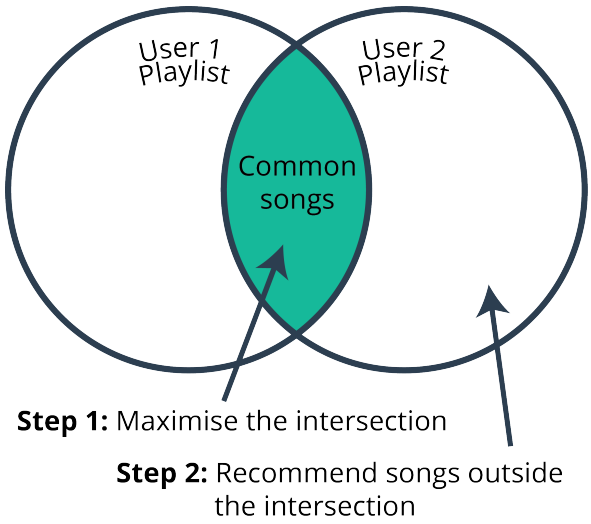

Figure 1: Similarity Venn Diagram

Since we will recommend according to the semantic information, it is necessary for us to retrieve the characteristics of some important characteristics about the data, such as top playlist titles, top album, top artists, etc. For example, these are what we have found in our explanatory data analysis about the top playlist titles:

- 10000 country
- 10000 chill
- 8493 rap
- 8481 workout
- 8146 oldies
- 8105 christmas
- 6848 rock

In our notebook demo, we will demonstrate a full statistic about characteristics we have found about the dataset.

3 Collaborative Filtering Model

To recommend items via the choice of other similar users, collaborative filtering technique has been proposed.

As one of the most successful approaches in recommendation systems, it assumes that if user X and Y rate n items similarly or have similar behavior, they will rate or act on other items similarly. Instead of calculating the similarity between items, a set of nearest-neighbor users for each user

whose past ratings have the strongest correlation are found. Therefore, scores for the unseen items are predicted based on a combination of the scores known from the nearest neighbors.

In our case, two playlists are similar if they have shared many of the same songs. The method used to recommend songs will be to recommend songs that are similar to songs already in a given playlist. This is shown using the Venn diagram in Figure 1.

3.1 Algorithm

First, to process our training set, we will create a dictionary that maps each playlists' pid which is unique to its track list. The reason to do this is that in the future it will be very convenient for us to locate all the tracks of a playlist given its pid.

Then, given a testing playlist which needs recommendation, we will first collect all the names of tracks, normalize them in a list, and then compare the list to all playlists in our training set, calculating the similarity score for all training playlists.

Assume for training list L, the number of songs shared between testing list and L be M, and the total number of songs in the testing list be T. The score function is calculated as follow:

$$S = M/T \tag{1}$$

We will calculate the score for all playlists in our training set and then rank the scores from high to low. Finally, we will recommend songs that are not in the testing playlist from the ranked playlists, where the most similar album will contribute first.

4 Word2Vec Model

Word2vec is a class of neural network models that were initially introduced for learning word embeddings that are highly useful for Natural Language Processing tasks. In recent years, the technique has also been applied to more general machine learning problems including product recommendations. The neural network takes in a large corpus of text, analyzes it, and for each word in the vocabulary, generates a vector of numbers that represent that word. Those vectors of numbers are what we are after, because as well see, they encode important information about the meaning of the word in relation to the context in which it appears.

if two different words largely appear in similar contexts, we expect that, given any one of those two words as input, the neural network will output very similar predictions as output. How does that relate to music recommendations? We can think of a users listening queue as a sentence, with each word in that sentence being a song that the user has listened to. So then, training the Word2vec model on those sentences essentially means that for each song the user has listened to in the past, were using the songs they have listened to before and after to teach our model that those songs somehow belong to the same context.

4.1 algorithm

There are mainly 2 algorithms that are used for training such vectors:

Skip-Gram: Skip-gram works well with small amount of the training data, represents well even rare words or phrases. The input to the model is wi, and the output that is predicted is wi1, wi2, wi+1, wi+2. This can be formalized as given the word and predicting the context.

CBOW (Continuous bag of Words): CBOW works several times faster to train than the skip-gram, slightly better accuracy for the frequent words. This works by giving the context to the model and predicting the center word. i.e. wi2, wi1, wi+1, wi+2 is fed to the model and wi is the output of the model. The number of words that we use as context depends on the window size that we define.

In this work, we will use Skip-gram because we expect the distribution of songs is very spread-out and the likelihood of a song appearing with high frequency is relatively low.

5 Experiment Setup

We will select the first 10000 playlist in our experiment. We will randomly split the data into 7500 playlists as our training set and 2500 playlists as our testing set. First, we will perform aggregated explanatory data analysis on the whole 10000 playlists, and then we will process to data to prepare for running our model such as building a dictionary mapping pid to tracks for our Collaborativer Filtering Algorithm. We will also use the 7500 training playlists to train our Word2Vec model. In our Word2Vec model, we set the vector size to be 150 in length, and we consider a window of size 10, which means only the nearest 10 neighbors are considered for a specific item at a time. Lastly, as illustrated in previous section, we are selecting the Skip-gram algorithm in order to address the spread-out distribution of songs.

We will evaluate our model with the 2500-playlist testing set. For each playlist in the testing set, we will first randomly partition the tracks of the playlist into 2 halves. The first half will be used to generated recommendations, and the second half will be withheld for evaluation. We decide to generate 5 recommendations for each track and use Jaccard Similarity as our metric. Jaccard Similarity measures intersection over union. In our case, Jaccard Similarity measures the number of correctly predicted songs over the total number of songs in the recommendation set plus the withheld set.

6 Result and Analysis

In our last, benchmark, we only evaluated the Collaborative Filtering model. In this benchmark, we focus on evaluating the Word2Vec model. Let the number of shared songs between withheld playlist and our recommendations be S, and let the length of original testing list be N, The Jaccard Similarity is calculated as follow:

$$JaccardSimilarity = S/3N \tag{2}$$

Since we are generating 5 recommendations for each track in the testing set, and we are withholding half of the testing set for evaluation, the denominator is (5*(N/2) + N/2) = 3N.

We evaluate the Jaccard Similarity in the first 10000 playlist dataset using Word2Vec model with different parameters as well as using the Collaborative Filtering model. Below is the average Jaccard Similarity we found:

- Collaborative Filtering model: 0.0235701
- Word2Vec Skip-gram: 0.0373758836981
- Word2Vec CBOW: 0.0278364723813

We have conducted one additional experiment for the best performing model: Word2Vec with Skip-gram. We scale the dataset by 10 times and evaluate the Jaccard Similarity for the first 100,000 playlist dataset and got a 0.785 Jaccard similarity. Thus, we believe the experiment results will be consistent as we scale the experimental dataset.

The result proved that Word2Vec produces superior results than simple Collaborative Filtering approach. Moreover, in the case of Song Recommendation, Skip-gram is better suited algorithm than CBOW. The reason behind the better performance lies with the nature of the two algorithms: CBOW works several times faster to train than the skip-gram, and provides slightly better accuracy for the frequent words, while Skip-gram works well with small amount of the training data, represents well even rare words or phrases. In our case, songs in the playlists are niche and scattered, and our experiment dataset has a relatively small size. Thus, Skip-gram model provides a better accuracy.

7 Future works

In the future work, we will improve the complexity of our Collaborative Filtering model by taking into consideration more complex attributes such as the shared album, artists as well as the number of playlist follower, etc. Moreover, we will also experiment with various parameter settings for the Word2Vec model in order to produce the optimal recommendation accuracy. It would also be interesting to combine the two model into a hybrid model and compare the results.

We are also interested in considering more extreme cases in music recommendation. For example: what if a playlist only contains one or two songs? what if a playlist contains no songs but only provides a name or a piece of text description? By looking into more semantic cues, we believe these problem could also be effectively addressed.