Automated Playlist Generation Using SVM Classification

Yunus Can ÇAKIR
2171452

Buğra Aker YILMAZ
2167617

Veli Özgür YILDIZ
2036366

***Abstract*—A new set of experiments for automated playlist generation problem using SVM is presented here. Based on audio features gathered via Spotify API, generated playlists allow users to discover new playlists they could enjoy listening to. The approach is based on the Support Vector Machines from machine learning and does not require any kind of metadata other than the release year of the tracks. It is evaluated using objective genre labels. It is validated with cross validation. Our approach allows people to create a new playlist based on their own playlist choices and music tastes.**

***Index Terms*—Playlist generation, SVM, Spotify API**

I. Introduction

There have been numerous papers and research for music playlist generation, based on previously listened tracks. This has been a hot topic significantly since such music platforms like Spotify and Fizy emerged, since these platforms provide a dataset on user profiles and audio files that was never existed before. In the literature, there have been various methods and algorithms to generate recommended songs; from naive classification algorithms to many machine learning algorithms such as ANN and RNN to clustering algorithms.

Recently, with the improvement of technology, many machine learning algorithms have become mainstream and one of these algorithms is Support Vector Machines(SVMs) due to its robust theory and efficiency, and its giving a confidence level besides classification label. In this paper, we present a playlist generation algorithm which uses SVM's as its kernel. In previous studies, it's seen that feature extraction of tracks is a very complex process. As Spotify offer such features of a track on the fly, we will be using their API for the feature and metadata of tracks as well as the source of the playlists.

II. Literature Review

The paper written by Bonniin ang Jannah [5] provides a good first insight on the subject, and summarizes the advantages and disadvantages well of the previously used methods. Below, we will inspect some of the previously used methods and some common problems on the subject.

In general, similarity based algorithms have been implemented[5]. For example, Ikeda et al.[9] proposed a solution using the distance between the last song and the possibly recommended songs. In their study, they presume that the order of the songs in a playlist make a difference. However Vall et al.[13] concluded that the order of the songs does not make a significant difference on the models used to recommend a song. They also concluded that context order, unlike song order does make a significant effect on the performance of the experimented models. Moreover, they observed that their most complex and successful models that uses RNN (Recurrent Neural Network), had similar performance with a simple popularity based algorithm. This shows that there is a general bias towards popular songs in an automated playlist generation system. To be able to cope this problem, we decided to not to include popularity as a feature when training a model. Also, since it does not make a significant difference on the performance, we decided not to take into account the song order unlike the studies [4], [6]. Other studies that uses similarity based algorithms are [4] and [7].

In another paper from Liu et al. [10], the classification takes the time of the day into account. For instance, each user may have different musical preferences depending on the clock, one may listen rock in the morning and slows at night. They create their own application for users to profile them in groups via surveys. Then, take the audio features and metadata from previously listened songs with the minutely timestamp to the account and generate new songs based on these features. They use a hybrid technique in which they first cluster the data using k-medoids, then use these clusters as the ANN inputs. For efficiency, they use 2 ANN's, namely long term ANN and a short term ANN. In their paper, they conclude that the time of the day is an important factor for the music selection.

In some other papers [11], start and end songs of a playlist were taken in the consideration. However, they concluded that the genre of music in a playlist have much more importance over start and end songs; thus, we agreed on not take these into consideration. For another paper, they dynamically generated playlists using skipping behaviour of the user. They used a combination of spectral similarity and fluctuation patterns (heuristic D). They concluded that heuristic D reduces the number of skips drastically. However, they stated that they did not use a filter for songs that are from the same artist and since heuristic D depends on similarity, a lot of songs from the same artist went into the same playlist.[12]

In this paper, we assume users have their own sets of playlists depending on the time of the day and the generation of new songs are to be done with recently played playlists.

III. Methodology

Our aim is to provide recommendations (some number of songs that are not listened by the user before) based on users' preferences on music. We had two main problems to solve. First, in order to provide such recommendation, a listening log or a an input that includes sample songs that user likes is necessary. However, there is no such public data. Therefore, we consider playlists as representatives of such data, i.e. we assume playlists are liked songs by the user. The method below could be applied to a listening log as well. Second, in order to provide N recommended songs, the model should be able to make a selection between candidate songs. As noted earlier in the literature review section, most models solve this based on the distance to the input songs. However, our approach is a classification model (SVM) and it uses the scores of the candidate songs, which are their distances to the margin to decide.

At the core of our approach is the audio specification-based music similarity measure. These specifications are gathered using Spotify API[1] in addition to gathering playlist data from Spotify. Specifications include acousticness, danceablity, loudness (as dB), speechiness, instrumentalness, liveness, valence, tempo and energy and all of the specifications are values that are already normalized within a range and calculated by Spotify. We used metadata (year) of the tracks in the playlist to enhance our features. We used 8 manually chosen genre, which are Pop, Rock, Metal, Electronic, Rap, Jazz, Soundtrack and Classical. We also used this genres in some of the experiments. If this classification cannot be done by hand, it is possible to use genre classification algorithms. Therefore, a track is represented by a 11 dimensional vector. Also, those features that are not in the 0-1 range are normalized so that the distance is calculated unbiased.

Deciding if the song should be recommended or not is a binary classification problem. When training the models, we used some part of the playlist labeled as 1, meaning it should be recommended, and used equal number of songs obtained through various methods in the experiments labeled as 0, meaning it should not be recommended. Testing our models, we only used the part of the playlist we hide from the model, so there is no 0 labeled data in the test part since there is no absolute knowledge about the songs that are not in the playlist, i.e. they could be in the playlist. All of our models uses RBF kernel and misclassification penalty parameter 1.0. We used accuracy as the performance metric, but as stated in the literature the performance metric of the models are highly controversial.

IV. Dataset

For the tracks, there are some datasets on the Internet[3]. However, these datasets provide no extra features other than metadata of the songs. Since this is the case, we selected 8 genre for the dataset selection and using Spotify API[1], we gathered 4 popular playlists for each genre, 2 from the playlists generated by Spotify and 2 from the users' public playlists. Each playlist consists of well over 50 songs. After the gathering of the playlists, we detected the duplicate songs with not only the name of the song but also the data of the song and deleted these duplicates. After getting over the duplicates, we used Spotify API to get the audio feature about each song:

- **Release Year:** Release year of the song, mapped linearly to 0-1 range.
- **Danceability:** How suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength and overall regularity. Between 0 and 1.
- **Energy:** Measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
- **Loudness:** Averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db. Mapped to 0-1 range linearly.
- **Mode:** Indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
- **Speechiness:** Detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
  Acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- **Instrumentalness:** Predicts whether a track contains no vocals. Ooh and aah sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly vocal. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
- **Liveness:** Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

- **Valence:** A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
- **Tempo:** The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration. Mapped to 0-1 range linearly.

  The above mentioned data was enough and we went on with experiments.

V. Experiments

A. *Different Techniques to Select 0 Labeled Training Data*

The first part of our experiments mainly consists of the techniques we tried to select the 0 labeled data to train our model. We split the dataset half and used one as the training data labeled 1 and other half as the test data. There were considerable differences between performance of the same method with different playlists. We provide mean, median, variance and standard deviation of the averages obtained from different playlists. Also, there are serious complexity difference between each method.

Technique 1

For the first technique we selected the 0 labeled data randomly from the songs not in the playlist. With this simple approach, we obtained a high accuracy which can be because of the fact that randomly chosen data represents the distribution well. The overall accuracy came out to be 91.58%

Mean	Median	Variance	Standard deviation
91.58	94.99	118.31	10.88

Technique 2

In the second technique we selected the 0 labeled data by selecting N number of samples from each genre, where N is (number of songs in a playlist) /2/numberOfGenres. With this approach we had a slight increase in the overall accuracy as the result was 92.24%

Mean	Median	Variance	Standard deviation
92.24	96.67	110.86	10.53

Technique 3

Unlike the second technique, in this technique we first determined the average minimum and maximum distances from the playlist data, then by dividing this data to 8, which is the number of genres, we get an interval for each of the distances. Then using this intervals we selected a random N number of songs, as 0 labeled instances, as described in the previous technique. In this technique, the overall accuracy came out to be 96.48%

Mean	Median	Variance	Standard deviation
96.48	100	38.60	6.21

Technique 4

In this technique, we selected the 0 labeled data to be the farthest distances from the playlist centroid. Using this approach, the overall accuracy came out to be 97.57%

Mean	Median	Variance	Standard deviation
97.57	100	21.15	4.60

Technique 5

In this technique, we selected top 3 nearest genres based on genre means and make a random selection from those samples as 0 labeled data. Using this approach, the overall accuracy came out to be 80.53%

Mean	Median	Variance	Standard deviation
80.53	86.93	315.17	17.75

Technique 6

In this technique, we calculated maximum distance of a song in a playlist to playlist centroid and used this as a radiance of the hyper-sphere centered at the centroid. Then, we choose the closest and outside samples to the sphere as the 0 labeled data. Using this approach, the overall accuracy came out to be 93.76%

Mean	Median	Variance	Standard deviation
93.76	95.83	33.77	5.81

Technique 7

In this technique, we calculated maximum distance of a song in a playlist to playlist centroid and used this as a radiance of the hyper-sphere centered at the centroid. Then, we choose randomly from the outside samples as the 0 labeled data. Using this approach, the overall accuracy came out to be 97.32%

Mean	Median	Variance	Standard deviation
97.32	100	15.66	3.96

B. *Cross Validation of Different Techniques*

After the first part of the experiments done, we chose the techniques with over 90% overall accuracy and put their models into 8 fold cross validations individually. The mean of these individual results of this cross validations are gathered to obtain the below figures. Here, we see that those techniques with poor representation of the 0 labeled data have failed and more advanced techniques obtained higher results up to 98

Technique	Mean	Median	Variance	Standard deviation
1	79.88	82.19	95.12	9.75
2	76.55	76.98	55.27	7.43
3	81.52	81.77	16.313	4.04
4	98.34	99.48	5.90	2.43
6	95.13	96.67	17.05	4.13
7	97.11	97.91	9.20	3.03

VI. Conclusion

In this study we examined an algorithm for SVM models to generate playlists based on previously listened songs. We provided several techniques for the training of the model and test them using 8-fold cross validation. It is problematic to test the output of playlist generation algorithms. We provided a method to test it, however, it tests only the tracks that are in the playlist. According to the results obtained by this test, the representation of songs not in the playlists plays an important role on the training. The best results were obtained from a technique where the songs not in the playlists are represented by the songs furthest away from the playlist centroid. The following two best techniques also uses the songs that are away from the centroid. By these techniques, we had an overall increase in accuracy of nearly 20% from the simple model that where the songs not in the playlists are selected randomly.

References

[1] Spotify API. Available at https://developer.spotify.com/documentation/web-api/

[2] Genius API. Available at https://docs.genius.com/

[3] Million Song Dataset. Available at https://labrosa.ee.columbia.edu/millionsong/

[4] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. *Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns.* In Proc. RecSys. 131138.

[5] G. Bonnin and D. Jannah. 2015. *Automated Generation of Music Playlists: Survey and Experiments.* ACM Computing Surveys (CSUR) 47 (2), 26

[6] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. 2012. *Playlist Prediction via Metric Embedding.* In Proc. KDD. 714722

[7] K. Lee and M. Cho. 2011. *Mood Classfication from Musical Audio Using User Group-Dependent Models.* In Proc. ICMLA. 130135.

[8] P. Hartono and R. Yoshitake. 2013. *Automatic Playlist Generation from Self-Organizing Music Map.* Journal of Signal Processing 17, 1 (2013), 1119

[9] S. IkedaMusic, K. Oku, K. Kawagoe. (date). *Playlist Recommender System AFT-IS* (journal)

[10] N. Liu, S. Hsieh.2010. *An intelligent music playlist generator based on the time parameter with artificial neural networks* (journal)

[11] A. Flexer, D. Schnitzer, M. Gasser, G. Widmer. 2008. *PLAYLIST GENERATION USING START AND END SONGS* (conference)

[12] E. Pampalk, T. Pohle, G. Widmer. 2005. *Dynamic Playlist Generation Based On Skipping Behavior*

[13] A. Vall, M. Quadrana, M. Schedl and G. Widmer. *The Importance of Song Context and Song Order in Automated Music Playlist Generation*