DS-GA 1013 Mini Project: Related Artists Recommendation

Jing Tao
jt3149@nyu.edu

Jianfei Xue
jx898@nyu.edu

Daiheng Zhang
dz2266@nyu.edu

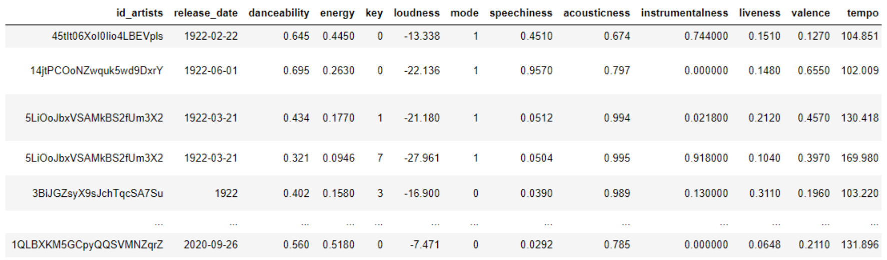
May 5 2022

1 Introduction

Recommendation system is a widely studied field of machine learning. There are three common ways to build a recommendation system: (1) collaborative filtering on user's ratings on items; (2) content based filtering which utilizes a profile of a user's preference; (3) hybrid methods. In this report, we would like to explore the possibility of building a recommendation system with very limited data (no user's ratings or user's preference profile). We build a recommendation system which recommends related musical artists given one artist. Hopefully, this can serve as an alternative way to alleviate the "cold start" problem which is when a recommendation system hasn't collected enough data to provide reasonable recommendations.

2 Data and Preprocessing

We use a dataset from one of the largest audio-streaming firm, Spotify. It contains over $600,000$ song tracks from 1921-2020[1]. The features include both song attributes and numerical attributes. Song attributes include the release date of the track, artist of the track, name of the track and etc. Numerical attributes are the aspects of the track measured in numerical ways. Figure 1 shows an overview of the track data. We have dancebility (how suitable a track is for dancing), energy (perceptual measure of intensity and activity), loudness (loudness in decibels(dB)), key (major key), speechiness (presence of spoken words), acousticness (how acoustic) instrumentalness (confidence of whether a track contains no vocals), liveness (presence of audience), valence (musical positiveness conveyed) and tempo (Beats Per Minute). All these metrics are generated by Spotify API and most of them, except loudness and tempo, are in the range of $[0, 1]$, a measurement of magnitude of the aspect.



id_artists	release_date	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo
45tIt06XoI0lio4LBEVpls	1922-02-22	0.645	0.4450	0	-13.338	1	0.4510	0.674	0.744000	0.1510	0.1270	104.851
14jtPCOoNZwquk5wd9DxrY	1922-06-01	0.695	0.2630	0	-22.136	1	0.9570	0.797	0.000000	0.1480	0.6550	102.009
5LiOoJbxVSAMkBS2fUm3X2	1922-03-21	0.434	0.1770	1	-21.180	1	0.0512	0.994	0.021800	0.2120	0.4570	130.418
5LiOoJbxVSAMkBS2fUm3X2	1922-03-21	0.321	0.0946	7	-27.961	1	0.0504	0.995	0.918000	0.1040	0.3970	169.980
3BiJGZsyX9sJchTqcSA7Su	1922	0.402	0.1580	3	-16.900	0	0.0390	0.989	0.130000	0.3110	0.1960	103.220
...	...	...	...	...	...	...	...	...	...	...	...	...
1QLBXKM5GCpyQQSVMNZqrZ	2020-09-26	0.560	0.5180	0	-7.471	0	0.0292	0.785	0.000000	0.0648	0.2110	131.896

Figure 1: Data in the form of DataFrame. Not all features are included in the figure due to space limitation.

We also have a dictionary containing related artists information: the related artists given one artist based on Spotify's user data. We treat these relations as ground-truth labels which we will try to predict as the right artists to recommend. We noticed that there are some artists that appear in the relations but have no tracks in the dataset, so we just remove them from the relation table. Also, we kept the 10 numerical attributes of tracks for further analysis. The resultant track dataset consists $586,000$ data samples of 10 dimension. For the relation table, we have over $600,000$ related artist pairs with $115,000$ unique artists.

3 Problem Formulation

With the true related artists pairs in the related artists table, we want to predict a list of most related artists for a given artist. However, neither do we have direct data on artists nor their listeners: the only data we have are track data with acoustic attributes for the song tracks produced by each artist. Hence, we want to formulate embedding representations for artists using the track attributes we have. Ideally, two related artists would have their embedding representations close to each other in the embedding space. Then we could apply K-Nearest Neighbors method to each artist embedding to find a list of other artists who have close embedding representations as our recommendation. Here, we choose K to be the number of related artists for a given artist in the true relation table.

4 Approaches

4.1 Artist Embedding

We have tried 5 approaches for artist embedding. The first and simplest one is simply taking the mean of attributes across all tracks for each artist and treat the resulting mean of each attribute as the representation for the artist. Here we assume that the overall characteristic of tracks made by each artist could represent its creator in some way. The second approach is similar to the first one, but after taking the mean, we apply PCA for dimensionality reduction. However, PCA does not performs very well since the variance explained by each eigenvector direction is almost even, resulting in small eigengaps. As a result, the dimensions only reduced from 10 to 8. The third approach is a little different: instead of averaging over all tracks of each artist, we treat each track as a distinct embedding representation of its artist. So each artist may have multiple embedding representations. For the rest two approaches, we assume that each artist may have produced tracks of different genres or styles and an artist may have different embedding in different genre domains. Thus we want to cluster the tracks of an artist into different groups first and then compute the mean attribute of each cluster as an embedding of artist. We apply two kinds of clustering method: K-Means clustering in approach 4 and Spectral clustering in approach 5. Depend on the number of clusters, each artist may have more than one embedding representation. For approach 1 and 2, where we only have one embedding representation for each artist, we can normally apply KNN to artist embedding data as described in section 3. For approach 3, 4 and 5, where each artist may have more than one embedding ($n \geq 1$ embedding), we will treat each distinct embedding as a data point and find K nearest neighbors (K defined in section 3) for each embedding of a given artist with respect to all the rest embedding points (resulting in a total of nK nearest points) and only choose the top K nearest points with their corresponding artists as recommendations.

4.2 Clustering

Here we will elaborate on how we perform the clustering methods mentioned in previous section. Especially, we will explain how we choose the optimal number of clusters for both K-Means and Spectral clustering when applying to the set of tracks of each artist. For K-Means, the procedure is simple: we test the number of clusters ranging from 2 to $min(10, N)$, where N is the number of data points in the set of tracks by one artist, and use the Silhouette score to find the optimal cluster number. For Spectral clustering, we compute the affinity matrix of the tracks of an artist using Gaussian kernel with Euclidean distance metric. We also apply local scaling based on the k nearest neighbors[3] when calculating the affinity matrix for better clustering results especially when the data includes multiple scales. Based on the Eigengap heuristic[2], we select the optimal number of clusters to be the value m such that maximizes the difference between consecutive eigenvalues of the Laplacian matrix.

5 Evaluation and Conclusion

We evaluated our 5 approaches on 5 independent test sets, each with 100 artists and a correspond average total of 1300 similar artists. Given the 100 artists one by one, we try to predict their similar artists and check if they are in the ground-truth labels. We count the average total number of correct predictions and results are shown in Figure 2.

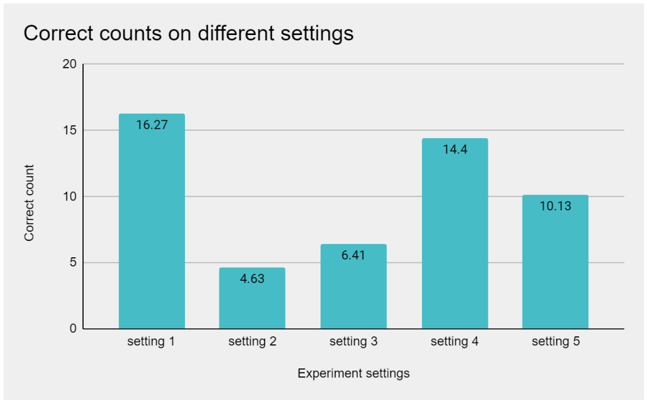


Figure 2: Evaluation results on 5 approaches/settings.

Our results shows a generally low level of precision, as in the best setting we only achieved 16 correct recommendations out of 1300. There are several reasons that might lead to this result. Firstly, the data contains artists and tracks from 100 years ago, while the user data is generated in recent year. It is safe to assume that listeners may has a bias towards looking for more recent tracks/artists than digging for the ones that has a history. Therefore, given an artist, the label we use might not accurately point to the artists that are most similar. Secondly, we know that KNN method suffers from curse of dimensionality. When the dimension of the data increases, the Euclidean distance between points become very steadily far meaning that there is no neighbors that are especially closer than others thus breaking the assumption underlying KNN. We have chosen 10 features which gives us a medium dimension scenario. Thus, the performance of KNN on this setting is questionable. Choosing different metrics for measuring distance/similarity might help (e.g. weighted Euclidean metric, cosine similarity). Another solve would be to apply algorithms that perform well on high dimension data such as Locality Sensitive Hashing.

References

[1] Yamac Eren Ay. "Spotify Dataset 1921-2020, 600k+ Tracks". In: (2020). URL: https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks.

[2] Ulrike von Luxburg. "A Tutorial on Spectral Clustering". In: *CoRR* abs/0711.0189 (2007). arXiv: 0711.0189. URL: http://arxiv.org/abs/0711.0189.

[3] Lihi Zelnik-manor and Pietro Perona. "Self-Tuning Spectral Clustering". In: 17 (2004). Ed. by L. Saul, Y. Weiss, and L. Bottou. URL: https://proceedings.neurips.cc/paper/2004/file/40173ea48d9567f1f393b20c855bb40b-Paper.pdf.