Authors: Andre Evard (Uni: ace2157), Kangrui Li (Uni: kl3350), Sofia De la Mora Tostado (Uni: sd3592), Yinqi Wang (Uni: yw4001), Zipei Jiang (Uni: zj2321)

Song Recommendation System for Spotify using collaborative filtering

Topics in Computer Science: Applied Machine Learning
Group: 1, Section: 013

For this project, we developed a model that recommends music to add to a playlist, given the previously added songs. Every listener has the constant challenge of finding the pieces that best suit their preferences or the mood of their playlists due to the sheer volume and accessibility of modern music platforms. Thus, to understand how these platforms present their recommendations, we created a few models to offer users customized recommendations.

We utilized the "Million Playlist Collection," a Spotify data set of playlists containing more than two million songs and more than 30,000 artists. The playlists of one million Spotify users between 2010 and 2017 were sampled and randomly distributed. For the original challenge, each of the 10,000 observations in the subsets of the data set in JSON format has some missing data. Going beyond the original challenge dataset, we gathered Spotify API audio features data and brief sentiment analysis using the Vader package, essentially, song metadata (AIcrowd, 2018).

The dataset was examined using statistics and graphs. We can view more than 2 million tracks with more than 280 thousand artists and more than 570 thousand albums from the 1 million playlists in this collection. See our second deliverable's slides for a summary of our data exploration.

Model

1. Collaborative Filtering

This technique is the process of filtering for information or patterns using collaboration among multiple agents, viewpoints, data sources, etc. This algorithm can organize the data into similar subsets and then makes recommendations for a given item based on the non-overlapping features of items in the same subset (Lee et al., 2012). So, for our dataset, collaborative filtering groups playlists based on their similarities.

2. K-Nearest Neighbors

In k-NN classification, the output is a class membership. An input is classified by a majority vote of its neighbors, with it being assigned to the class that is the most common among its *k* nearest neighbors (Papu, 2022). We implemented a K-Nearest Neighbors algorithm in our system to identify the K playlists with the smallest distance to a given playlist. This algorithm would return K playlists, which we could pull tracks from to recommend non-overlapping songs.

3. How to Implement

Our first step in implementing this algorithm is to create a new data structure to organize the playlists into similar subsets. We tried two approaches. Firstly, we use the one-hot encoding to create the sparse matrix based on whether the song is in the playlist. Every playlist can be encoded as a vector with length equal to the number of tracks in the dataset. In addition, we also tried to use Spotify audio features data for vectorization. We calculated the average score of eleven audio features for each playlist for all songs as the playlist vector.

Then we calculate the similarity between a given playlist and others based on two vectors separately. We tried the Euclidean and cosine distances for the metric to compute the distance (IBM). The result of the two metrics are similar, and we decided to use the cosine similarity in our model, which is given below:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

This similarity allows us to use the K-Nearest Neighbors to identify K playlists with the smallest distance to a given playlist. To generate a recommendation, we weigh each occurrence of a song in a K-nearest playlist according to the playlist's relative distance to the given playlist and then sum over all of these weighted occurrences:

$$Weight(t) = \sum_{i=0}^{K} i^{-d} * 1(t \in playlist_i)$$

Where the parameter d decides how fast the weight decay as the distance increases. We return the top n songs with the highest weights not in the input playlist as the recommendation.

4. Optimization and evaluation

To evaluate the performance of our model, we randomly remove a fraction of the playlist from the test set, pass the partial playlist to the model, and calculate the accuracy as to how well the recommendation recovers the removed fraction of the playlist. That is, the number of songs in both the removed fraction and the recommendation, divided by the size of the removed fraction:

$$Accuracy = |Removed\ List \cap Recommendation|/|Removed\ List|$$

Two essential parameters that need to be optimized are k, the number of nearest neighbors to be returned for the recommendation, and d, the decay regularizer to compute weights of recommended songs. And we did several experiments corresponding to different parameters, and here is the recommendation accuracy of them (d for columns and k for raws):

	10	20	30	40	50	60	70	80	90	100
0	0.259	0.342	0.416	0.399	0.419	0.485	0.451	0.460	0.443	0.469
0.5	0.214	0.414	0.336	0.396	0.381	0.421	0.532	0.473	0.389	0.440
1	0.292	0.311	0.359	0.375	0.379	0.374	0.314	0.410	0.368	0.448
2	0.303	0.395	0.439	0.415	0.402	0.321	0.377	0.399	0.393	0.361

From the results, we find that as the number of neighbors is 70. The decay parameter is 0.5. We can achieve the best performance with 53.2% accuracy.

We also tried to combine the results from two vectorization approaches. That is, compute the recommended list according to the nearest k/2 nearest neighbors of the sparse one-hot encoded matrix plus the nearest k/2 nearest neighbors of the audio feature matrix. Using k=30, d=1, the sparse matrix using one-hot encoding recovers around 50% of songs removed from the playlist. And the audio feature matrix recovers 30% of removed songs. Considering the uncertainty of recommendations in the real world, our collaborative filtering model returns reasonable recommendations.

Feature Importance

In addition to the primary clustering mechanism, we assembled a secondary class of clustering recommendation models. Digging a little deeper into the dataset and polling from Spotify's API, there are two things that particularly caught our interest: the track_artist label and the audio features of every song. We combined these two further metrics, using small selections of the audio features (for instance, tempo and song duration) and an L2 error to calculate how close every outside track by a present artist is. Across every metric combination, two proximity calculations were generated: one that scored based on each song's closest song in the playlist and one that compared to the playlist summary. To evaluate how each metric and proximity approach performed, we masked a host of songs from each test playlist and determined how strongly they were recommended out of all recommended songs. In scoring, the track-based calculator significantly outperformed the summary one–however, that may be because the masking process disproportionately disrupted the playlist summary.

We also analyzed our model using cross-analysis and visualizations (Please see Appendix for the visualizations). Looking at the median performance by tracks and summary, though no recommender was incredibly accurate, we can determine the rough importance by observing the frequency among the best metric combinations. The most frequent included danceability, tempo, speechiness, and mode. We then did a combined analysis to select the features with good median and best performance scores with a lower worst performance score, and we found one top feature (danceability-mode), suggesting these two features, or the combination thereof, may be particularly worthy of future analysis. Each approach (summary and tracks) individually resulted similarly.

In short, the "mode", "danceability", "speechiness", and "tempo" features were the most important when using this approach of feature-based extraction. Were we to iterate on this model further, we would first experiment with different distance calculations than the L2 norm. Perhaps a regressor.

Results

We curated a selection of clustering models for recommending tracks. Their performance is reasonable, given the difficulty of the problem at hand, and the recommendation implementation process is much clearer to us now. The suite of our models could be combined into a single "model", in that every model generates a list of outputs that can easily be merged. They can present various plausible songs to add to a given playlist.

References

[1] Terveen, Loren; Hill, Will (2001). "Beyond Recommender Systems: Helping People Help Each Other" . Addison-Wesley. p. 6. Retrieved 16 January 2012.

[2] D. F. Murad, R. Hassan, B. D. Wijanarko, R. Leandros and S. A. Murad, "Evaluation of Hybrid Collaborative Filtering Approach with Context-Sensitive Recommendation System," 2022 7th International Conference on Business and Industrial Research (ICBIR), 2022, pp. 7-12, doi: 10.1109/ICBIR54589.2022.9786506.

[3] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[4] J. Singh, "Collaborative Filtering based Hybrid Music Recommendation System," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 2020, pp. 186-190, doi: 10.1109/ICISS49785.2020.9315913.

[5] IBM. “What Is the K-Nearest Neighbors Algorithm? | IBM.” Www.ibm.com, www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or.

[6] Lee, Joonseok, et al. “A Comparative Study of Collaborative Filtering Algorithms.” ArXiv.org, 2012, arxiv.org/abs/1205.3193. Accessed 02 November 2022.

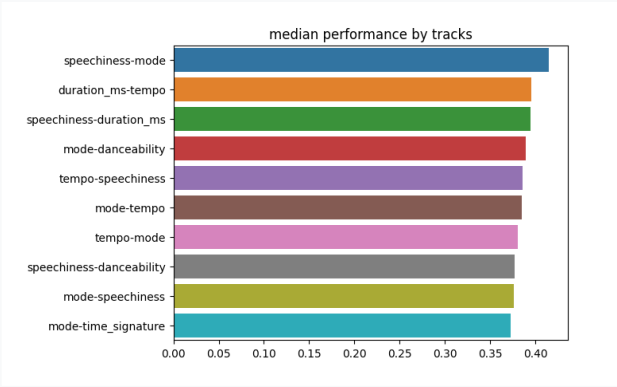
[7] Papu, V. (2022, October 14). W4995 Applied Machine Learning Fall 2022 Lecture 2 [PowerPoint slides]. Columbia University.

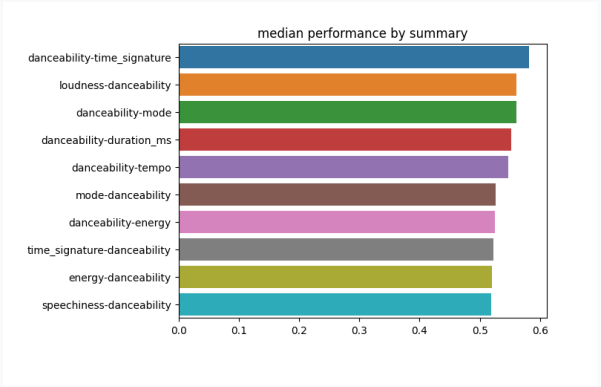
[8] Spotify. (2018). *Spotify Million Playlist Dataset Challenge: Challenges*. AIcrowd. Retrieved October, 2022, from https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge

Authors: Andre Evard (Uni: ace2157), Kangrui Li (Uni: kl3350), Sofia De la Mora Tostado (Uni: sd3592), Yinqi Wang (Uni: yw4001), Zipei Jiang (Uni: zj2321)

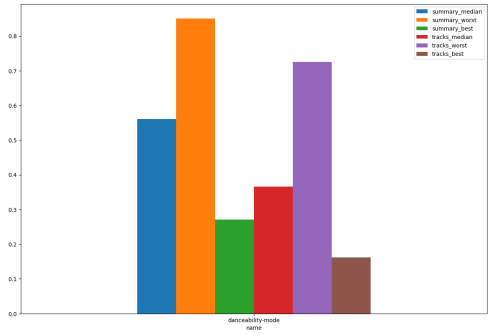
Appendix

Visualizations for the median performance scores by tracks and summary:





The top feature we found:



The visualizations of our analysis of each approach (summary and tracks) individually and the top features we found(the left one for summary approach and the right one for the tracks approach), we can see that tempo, energy, mode, and danceability are all present: