Music Discovery and Recommendation System

Team 101 - Akshat Chauhan, Nicholas Anthony Ghinazzi, Anupam Priyadarshi, Nathaniel Meng Pung, Daniel William Remmes, Anh L Tran

Motivation and Problem Definition

Our motivation for making this app was to develop a method for users to more fully control the recommendations from music streaming applications. Currently, music apps curate songs the user hears through a black box set of algorithms. We wanted to create something where the user has a lot more control over what they are offered. We wanted to choose something that could have a use in the real world, as well as something that we could use to show our skills gained in this class.

With the advent of Spotify, Pandora, Youtube and iTunes streaming services have contributed the majority of music industry revenue. Each of them provides their own methods of recommendation, but none allow for the detailed customization for a playlist that is desired. We would like to develop an app that gives the users specific control over what characteristics are driving their playlist and which artists they see.

Survey

In earlier times, online music curation relied on music experts or opinion leaders to create playlists for users. These playlists were created based on the perceived expertise with local touch. However, such methods lacked personal taste and later, a lot of online music companies started using Machine Learning techniques with different models for music recommendation systems with enhanced personalization and taste. Some of the commonly used techniques are –

- Collaborative Filtering models – used by Last.fm, Spotify, Netflix[4,5,17,18]
- Natural Language Processing (NLP) models, which work by analyzing text [3,16].
- Audio models, which work by analyzing the raw audio tracks themselves[12,13].

Most of the recommendation systems and studies that were done use audio models, lyrics and context data to classify the songs before recommending them to the users [7,8,9]. For example, natural language processing and sentiment analysis are often used to dissect and analyze the content, the semantic features of lyrics [2,10,11]. Plus, the audio models provide many useful attributes for improving performances of classification, especially in combination with lyrics [1]. Some of the recommendation systems proposed Collaborative Filtering and Genetic Algorithm[4], Convolutional Neural Networks[6,15] approach to classify the music and then Collaborative Filtering to provide recommendations and a distance metric learning algorithm [5] was applied for genre classification. Traditional classifiers like SVN or KNN were reported to reduce efficiency when applied on complex data [6].

Although many current music recommendation systems are helpful, they are mostly trying to find similar songs and do not allow the users to change what level of similarities they currently want. There are also other studies and challenges that try to predict what songs will be in a playlist (Million Song Dataset Challenge), but these are also based on similar

characteristics of each song [14]. A user is still forced to only discover songs based on the similarities of a current song they like.

For this project, we will attempt to create a user driven recommendation and artist discovery interface, an app that will allow the users to control each level of input to discover new songs. For example, a user will be able to choose various characteristics from a given song and create a playlist revolving around those that were selected. This customization includes many musical features such as liveliness, tempo, mood, energy, etc. that influence the listeners' preferences and predict what they might like. We can leverage previous studies and analysis methods to assign a score/classify each song.

Notably, in our method, we will not be using the listening history to make recommendations. Instead, we will be allowing the user to define specific characteristics of songs that they would like to listen to. Although our approach may not initially improve song recommendations, it will allow the user to explore other types of music they may not have heard of before. For example, if a user likes a specific song's key, loudness, tempo, etc. they can create a playlist surrounding those exact qualities, something that has not been done, or potentially discover a new artist that they will enjoy. The interface will be intuitive to allow all music lovers to generate customized playlists and to dive deeper into the pool of artists and songs that they may not be familiar with. This frees the users from having to discover new music based on a current one they are aware of. It will allow the users to change their preferences (and thus, the algorithm) as they listen to each new track, something not currently in the marketplace. We are exploring other ways of discovering music besides recommendations based on a single song or artist.

Our target users include everyone who likes music and wants to explore their music tastes. They will be able to use our system to discover new songs and artists. If we are successful in developing our recommendation and discovery system, current popular streaming music services may adopt some of our methods. In addition, it may highlight smaller artists who would not otherwise have a chance to reach a larger market.

Proposed Method

I. Intuition

We feel that our proposed methods are better than the existing song recommendation platforms because of the level of feature personalization we will be able to achieve. Due to its strict adherence to only filtering based on song parameters, this system will introduce users to new songs and different genres that they may not have been familiar with. It will serve to broaden the users comfort zone while still providing the key characteristics that they were looking for in a song.

II. Source Data

For our source data, we scraped the Spotify API for each year using Python. We chose Spotify since they bought out The Echo Nest company which was responsible for creating the Million Song Dataset. There were some limitations to get around on the API as each API call was limited to 2,000 maximum return.We got around this limitation by first getting the 2,000 albums for each year from 1950 to 2020. Then we grabbed each track from the albums. Therefore there is an implicit bias on our initial source since Spotify API will return the

most popular 2,000 albums for each year. However, this does not mean that every song in the album is popular. One thing about collecting and researching our dataset that was interesting is that we were faced with the decision about what to do with non-english songs. We decided to include them in the dataset as this would help us evaluate our algorithms and avoid troublesome outliers such as classical music and english songs from non-english locales. Duplicate songs with the same song name and artist name were removed. For the toy database, random songs were removed until a workable size was met (~40MB). Edge cases in the toy database forced us to remove tempo=0 due to uneven distribution of the value (~200 records). We also removed [Instrumentalness] from the website due to uneven distribution as well (⅓ of all records were 0).

III. Algorithm

In order to determine what the best songs to recommend, a weighted clustering algorithm is employed. The user will provide a song and the features that they would like to base a playlist around, liveness, tempo, etc. Using this information, after normalization, the original dataset is reduced to a localized space around the point which was provided. By reducing the dataset to a smaller volume centered at the given point, we are assuring that the songs suggested will resemble the song and parameters requested. Within this new space, clusters of equal weight are created.

A list of 10 songs is generated by first selecting a cluster randomly based on the supplied weighting. Once a cluster has been selected, a random song from within the cluster is chosen to add to the list. This list of songs is provided to the user to either approve or disapprove.

As each group of 10 songs are judged, the weight of each cluster is updated. For example, if a set of three clusters was used, and evenly weighted, they would start with scores [10, 10, 10]. Songs from the list which receive a “thumbs up” add a +1 to the value of the associated cluster, those which receive a “thumbs down” will subtract 1 from the associated cluster. With the following judgements applied to each cluster, (0: [Up, Up, Up, Down], 1: [Down, Up, Up, Down], 2: [Down, Down, Down]), the resulting cluster weights would be [12, 10, 7].

This new cluster weighting would then influence the selection of the next 10 songs which are provided to the user for judgement. By varying the initial weights applied to each cluster, and the number of clusters, it is possible to adjust the rate at which the cluster scores reach relative stability.

IV. User Interface

Before the users can use this application, they need to log in to it. We store the user data in sqlite db for authentication and once authenticated that user data is stored in session. For the testing purpose, we provide a username and a password for the users at this time. In the future, we will update this app with the user registration.

In the homepage, the user interface encompasses five separate sections illustrated in an accordion style. The five sections are: Discover, Explore, Visualize, About, and Log out. Each section has its own brief description explaining its purposes.

In the “Discover” section, the user can search for a song of their choice first and select either all or some musical features such as tempo, energy, liveness, valence, danceability,

speechiness, acousticness, and instrumentalness. Then the user can click on the “Discover Songs” button. After that, the system will generate clusters and offer up songs from each cluster and list them in a table format. Another function is that the user can select the song(s) they like or dislike by clicking on the thumb-up or thumb-down icons in the likability column. Then the system will generate a new list of 10 songs based on those likable songs. A sample of the song is available to experience a snippet of the song. We have also provided a link to forward the user to their Spotify app to enjoy the full song and add it to their playlists/favorites (must have a Spotify account). We chose not to implement these functions within the website due to security/privacy concerns, quality, user experience and time constraints.

In addition, once the user has reached a point that they are satisfied with the music suggestions, they have the option to “View Cluster”. This allows the user to visualize which clusters they have prioritized over others. A 3D scatter plot will show their selections displayed in relation to each other, denoting their preferences in order. Also, a radar chart takes the same information and overlays the parameters of each cluster to let the user interactively compare the clusters. They can hide, rotate, and see values of each.

In the “Explore” section, the user can explore the entire dataset by selecting the song attributes that they care about. There are six different attributes to filter on (i.e. year, temp, energy, mood, liveness, and danceability), and a search function that automatically filters the results as the users type. A sample can also be played (dependent on Spotify availability).

In the “Visualize” section, there is an interactive dashboard of the source data. If the “Explore” part provides details about the data, this part gives the user a new perspective to observe our data being aggregated such as the total numbers of songs, artists, and albums. Plus, it also provides a bar chart of the top 10 most popular genres based on number of records and a bubble chart of the top 20 artists based on popularity scores. The user can also select the timeframe of the data by year(s) and enter a genre by which the dashboard will update automatically.

In the “About” section, it includes information about our projects and team members.

To log out of the system, the users can click on “Log out” section.

Experimentation/Evaluation

After we have trained the database we will experiment to evaluate our algorithm. For our experimentation we will have users input songs and interact with the results. We will use feedback from these experiments to refine our algorithms to obtain better results. We will initially tailor the results based on euclidean distance from the input song or variables, but we will modify this if fine tuning of results is found to be necessary.

- Filtering Performance: To determine where filtering the data makes the most sense, a quick study was performed by altering the location of filtering the dataset. Each method started with the full dataset, 1.57 million rows, filtered by artist_popularity > 70, and then an additional filter around the parameters provided by the user. This brings the final dataset down to 8,540 rows in each situation, filtering on the SQL import and filtering within python. The results are below, it only took 10 rounds of filtering to show that filtering at the SQL import is significantly better.

Filter Location	*Time*
SQL	**1.75**
Python	6.90

- Clustering Evaluation: An optimization study was performed to determine what number of clusters would be best to ensure timely convergence of recommendation, defined as one cluster doubling in size, indicating a dominant cluster liked by the user. Using the same algorithm described above, a simulation was run through 100 times for each number of clusters, 4 through 8. Each time, the individual clusters were given a random rate at which the user would "thumbs up" the song requested from the cluster, allowing for 50 rounds of feedback to converge. As it can be seen below, a 7 cluster algorithm performed the best.
-

Starting Score	*4 Clusters*	*5 Clusters*	*6 Clusters*	***7 Clusters***	*8 Clusters*
5	9.19	7.69	8.36	7.43	7.08
6	10.31	10.28	9.11	8.83	10.02
7	13.78	10.99	10.54	9.72	10.59
8	15.87	11.17	12.15	12.07	12.07
9	15.85	13.41	12.66	13.34	16.11
10	16.74	16.36	15.03	14.41	15.53
Avg Convergence	*13.62*	*11.65*	*11.31*	***10.97***	*11.90*

- Number of features: Songs were randomly selected and put through a series of filters, each with an increasing number of parameters included. Each started with 1.57 million rows, and after the filters the resulting dataset rows and percent of original dataset are shown below.

Features	*Filtered Dataset*	*% of Original Data*
1	889,939	56.57%
2	396,171	25.18%
3	***161,810***	***10.29%***
4	36,327	2.31%
5	13,041	0.83%

From the results, we chose to limit selection to three features to filter on. With three features, the user has a significant number of combinations that can be made,

but beyond three, we are risking working with too small of a dataset, resulting in subpar recommendation.

- Relationships between artists: Create a graph of artists that have toured / performed together. This data was taken from musicbrainz.org [19] which catalogues music information from user submitted data. The list is not comprehensive but it is a source that is free and open for everyone to use and has a substantial amount of data. We were originally planning to use this data to integrate into our song selection algorithm. However, we were unable to merge this successfully. The data itself is interesting to see. The file is located in the data folder - artist_relationships.tsv. We loaded the data into argolite and saved a snapshot - artist_relationship_argolite_snapshot.txt [20]. Please note that it takes about 3GB of RAM to load this node/edge graph. Some interesting tidbits from the graph - Bruce Springsteen, KISS, Alice are among the top performers who have toured with many bands.
- Music genres : The dataset represents songs from almost all corners of the world hence a network graph is created to represent a linkage between 3k+ musical genres . This graph can be used to analyse as to what extent a particular genre such as"Rock" has evolved into (for eg 200+ offshoots) over the period of 70 years. This shall help users explore and discover the extent of localized influences over individual music genres.

Throughout the development phase, we consistently tested and re-tested user interactivity and run times of the web site/app as a whole. We tweaked and continuously fixed bugs to improve user experience. There are places that we can improve upon, like user creation and updating the algorithm to integrate the like/dislike table for keeping track of users preferences for multiple sessions.

Conclusion

Our application delivers on our stated goal of providing the user with a new way of generating a playlist. Our novel approach places the control of the algorithm in the hands of the user, rather than the results being generated from a "Black Box" that the user has no control over. We used a combination of technologies, all open source, to develop our application, and used the skills learned in this class to develop transformations and visualizations that bring the data to life, for the user.

We faced a number of challenges during development. We first had the challenge of obtaining the data. We had to come up with a way to get around the 2000 song limit on the Spotify API. We next faced the challenge of how to make the UI usable for the end user and still allow them the freedom we were looking for to create their own playlist. We came up with a multi-pronged approach that let the user choose from three different styles of list generation so the user is not overwhelmed, but still allows the user the maximum number of options to choose from. We enjoyed this challenge and gained further insight into the difficult and exciting world of software development, data analysis and visualization.

All team members have contributed similar amount of effort

References

1. Lee, J., & Lee, J. S. (2018). Music popularity: Metrics, characteristics, and audio-based prediction. IEEE Transactions on Multimedia, 20(11), 3173-3182.
2. Napier, K., & Shamir, L. (2018). Quantitative Sentiment Analysis of Lyrics in Popular Music. Journal of Popular Music Studies, 30(4), 161-176.
3. Hu, X., Downie, J. S., & Ehmann, A. F. (2009). Lyric text mining in music mood classification. American music, 183(5,049), 2-209.
4. Kim, H. T., Kim, E., Lee, J. H., & Ahn, C. W. (2010, April). A recommender system based on genetic algorithm for music data. In 2010 2nd International Conference on Computer Engineering and Technology (Vol. 6, pp. V6-414). IEEE.
5. Lee, J., Shin, S., Jang, D., Jang, S. J., & Yoon, K. (2015, January). Music recommendation system based on usage history and automatic genre classification. In Consumer Electronics (ICCE), 2015 IEEE International Conference on (pp. 134-135). IEEE
6. Chang, S. H., Abdul, A., Chen, J., & Liao, H. Y. (2018, April). A personalized music recommendation system using convolutional neural networks approach. In 2018 IEEE International Conference on Applied System Invention (ICASI)(pp. 47-49). IEEE.
7. Peter Knees and Markus Schedl. 2013. A survey of music similarity and recommendation from music context data. ACM Trans. Multimedia Comput. Commun. Appl. 10, 1, Article 2 (December 2013), 21 pages. DOI:https://doi.org/10.1145/2542205.2542206
8. Kahyun Choi, Jin Ha Lee, Xiao Hu, and J. Stephen Downie. 2016. Music subject classification based on lyrics and user interpretations. In Proceedings of the 79th ASIS&T Annual Meeting: Creating Knowledge, Enhancing Lives through Information & Technology (ASIST '16). American Society for Information Science, USA, Article 41, 1–10.
9. C. Laurier, J. Grivolla and P. Herrera, "Multimodal Music Mood Classification Using Audio and Lyrics," 2008 Seventh International Conference on Machine Learning and Applications, San Diego, CA, 2008, pp. 688-693.
10. Hu, X., & Downie, J. S. (2010, August). When Lyrics Outperform Audio for Music Mood Classification: A Feature Analysis. In *ISMIR* (pp. 619-624).
11. Hu, Xiao & Downie, J.. (2010). When Lyrics Outperform Audio for Music Mood Classification: A Feature Analysis.. 619-624.
12. Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music recommendation by unified hypergraph: combining social media information and music content. In Proceedings of the 18th ACM international conference on Multimedia (MM '10). Association for Computing Machinery, New York, NY, USA, 391–400. DOI:https://doi.org/10.1145/1873951.1874005
13. Sergey Volokhin and Eugene Agichtein. 2018. Towards Intent-Aware Contextual Music Recommendation: Initial Experiments. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18). Association for Computing Machinery, New York, NY, USA, 1045–1048. DOI:https://doi.org/10.1145/3209978.3210154

14. Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.
15. Audio-based Music Classification with a pretrained convolutional network, S. Dieleman, P. Brakel and B. Schrauwen, ISMIR '11
16. Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60).
17. Aiolli, A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge Preference Learning: Problems and Applications in AI (PL-12), ECAI-12 Workshop, Montpellier
18. Mukund Deshpande and George Karypis, 'Item-based top-n recommendation algorithms', ACM Trans. Inf. Syst., 22(1), 143–177, (2004).
19. MusicBrainz - The Open Music Encyclopedia. (n.d.). Retrieved April 3, 2020, from https://musicbrainz.org/
20. Li, Siwei, et al. *Argo Lite*, 2020, poloclub.github.io/argo-graph-lite/.

Appendix

This section will serve to mainly describe the technical details of obtaining the source data for node/edge graphs.

I.Artist to Artist Relationship Graph

The artist to artist relationship graph shows the number of times each artist has performed together before at an event. These events were taken from the musicbrainz.org(mb) open sourced database. Users of the website save events information for each particular artist. To grab the full dataset go to the website and then under Products->MusicBrainz Database->Download->Go to USA link ->select latest folder date -> download mbdump.tar.bz2. Unzip and the source files (event, l_artist_event, artist) should be there. The code file *CODE/artist_relationship.py* uses the spotify.db database to get unique artists then joins back to the artist dataset. This reduces all artists down to just the ones we have. Next we take all events from the event database (l_artist_event) and join it back to the artists. After this step, we find all artists at the same event and link them to each other. The event dataset is only used for the event name, it is not needed for the relationship graph but good for general information during debugging.

We were not able to incorporate this data into our algorithm or the webpage but felt this would be another step in the right direction in fine tuning it. We could display the artists the user liked/disliked so far and if they have had any performances together. This could also be used to tweak the song selection algorithm.

II. Artist Genre Graph

Each spotify artist contained a list of genres as well. This graph shows how many times each genre was grouped together within this list. Please see the code file *CODE/graph_analysis.py* for more information.