Spotify Playlist Challenge

Project Final Report

Brendan Ryan
University of Colorado Boulder
Boulder Colorado USA
brry3186@colorado.edu

Christopher Lescinskas
University of Colorado Boulder
Boulder Colorado USA
chle8754@colorado.edu

Matthew Martin
University of Colorado Boulder
Boulder Colorado USA
mama4085@colorado.edu

ABSTRACT

For this project we are looking to see if there are good ways to provide a playlist extension to an initial prompt using some basic data mining techniques. This type of analysis is incredibly valuable to organizations like Spotify that charge users a fee to stay on their platforms. To be certain, users get some value from being able to access an extremely large collection of music, but what keeps them engaged in these platforms is the quality of their engagement with them. If they are presented with playlists that keep them engaged, they are more likely to continue to remain paying subscribers of the service. We will dive more into the motivations for this type of research in the Introduction of this paper.

For this project we looked at the data in a few different ways. We wanted to evaluate different means of looking at the data and practice some datamining techniques on a large dataset. We are focused primarily in this evaluation on the correlation of songs to one another. We are not providing an analysis of the causality of song selection to other songs.

We started with some basic heuristics to understand things like the size of the dataset (1,000,000 playlists), the frequency of unique songs (HUMBLE. by Kendrick Lamar had the highest frequency at 46,574 instances), and the average playlist length (66 songs)

We started with the hypothesis that we could evaluate playlist extensions simply by looking at song titles in relation to one another. To get an early validation of this approach we conducted basic clustering analysis to see how similar (or dissimilar) the data was. This proved to indicate that there were likely many songs which would fall into dense clusters allowing some of our other analysis to proceed.

We wanted to start our analysis by performing an Apriori evaluation of the data. We knew this had the potential to yield good results but also understood that it might be a computationally heavy algorithm. Using a single challenge from the challenge dataset provided by Spotify (1,000 tracks with 5 songs each), we were able to generate 1000 thousand new playlists with 500 songs in each and no duplicate tracks within a playlist. One additional interesting result was that even with only chunking the dataset to 50,000 playlists at a time for Apriori analysis, runtime still took about 10-20 hours.

We also wanted to see how the results might differ with a K-Nearest Neighbors evaluation of the data. We knew that some of the approaches to this challenge had employed this technique and were curious to see what results it might yield. Unfortunately, we found ourselves a little stuck in understanding how to utilize this technique on non-numerical attributes, and decided to pivot to other methods for another approach.

We turned to Natural Language Processing as an alternative to the Apriori approach. This would take a slightly different approach to filling out a playlist by taking a prompt, such as “run” (as in a playlist that might be consumed while running for sport) or “rap” (a music genre), and returning an associated playlist. This was adapted from a similar approach that Ludwig et al.[4] took for part of their approach. This produced a result in considerably less time that appeared to meet expectations though presumably at a lower supportability or confidence level.

1 Introduction

There is a great deal of value in discovering patterns of like users for platforms like Spotify. The ability to provide suggestions to users and auto-generate playlists, with a high rate of user acceptance, provides stickiness and value to the user. The more value that the user experiences from this kind of app, the more likely they are to continue to pay for service and recommend to other users. According to Digital Media Association's Annual Music Report (March 2018)

> "54% of music consumers report that playlists are replacing albums for them."[1]

With this level of engagement, providing playlist with high user acceptance is vital to maintaining engagement with users and staying relevant in the digital music delivery space.

In this project, we imagine ourselves in the role of the streaming music provider, seeking to improve engagement by providing a feature to suggest new songs or artists for a user to listen to. Our primary goal is to provide suggested extensions to a playlist prompt. We are also curious about trends that may be discernable within the data, such as whether a song tends to appear at the beginning or end of a playlist, the likelihood for two songs to appear on the same playlist, or the degree to which a playlist has varied albums or artists will also be explored.

2 Related Work

Ludwig et al. (2018) describe a method for music recommendation using the MPD that combines multiple analyses into a hybrid model. The team used Item-Based Collaborative Filtering (ITEM-CF), Session-based Nearest Neighbors (S-KNN), and other analyses to generate initial models for recommendation. Models were then combined using techniques such as filling, weighting and switching. Results and effectiveness calculations are given for all base and hybrid models.

The Ludwig et al. paper argues that although computational complex algorithms can produce better results, those results can be only marginally better and often require very computationally heavy algorithms that may require specialized hardware to facilitate. That it may be a better approach to take get a very good answer, with less computational lifting, and therefore less resources and time required to produce results.

The team looked at three approaches to their hybrid approach including *Filling, Weighted,* and *Switching. Filling* involved filling out a playlist with the extensions of the playlists containing the prompt in order of frequency. When the results returned a playlist with less than 500 entries (the amount required by the challenge), they would fill the list out with the most popular songs. *Weighted* involved combining the results of multiple lists by ranking scores. *Switching* varied the method based off the seed input provided.

Kelen et al. (2018) also submitted to the 2018 RecSys challenge using the MPD. Their approach focused on k-nearest-neighbor analysis to recommend additional tracks. The team then used different methods to improve their model such as amplification, weighting by inverse item frequency and weighting by item position.

The *Efficient K-NN for Playlist Continuation* follows the same philosophy as the Ludwig et al. paper in that it attempts to produce a solution that does not require significant computational lifting or specialized equipment.

> "One major advantage of our approach is its low computational resource use: our final solution can be computed on a traditional desktop computer within an hour"[6]

Their approach uses a K-NN approach to determine the degree to which playlists are similar. A playlist that calculates as similar to the prompt is amplified to make it more important in the computation. The approach also takes into account how frequent a data item is. Songs that appear less frequently but appear on two lists are more a more valuable indication of similarity

than two song that are super frequent and therefore are likely to show on many playlist. Lastly, it looks at positional information of tracks. Assuming that if you are given a prompt as the first *k* number of tracks that a search for similarity should be concerned more with *k*+1 to *n* tracks rather than tracks 1 to *k*.

As both papers are submissions to the 2018 RecSys challenge, they are directly related to our work and demonstrate the work that has been done thus far on the dataset.

3 Data Set

3.1 Dataset Basic Statistics

The dataset being used for the analysis discussed herein is provided by AIcrowd for the *Spotify Million Playlist Dataset Challenge.* This data set consists of

> "… public playlists created by US Spotify users between January 2010 and November 2017."[2]

It was provided as part of a public data mining challenge that is a continuation of a challenge that originally ran in 2018.

The following is an initial look at the dataset provided for the Spotify Playlist Challenge.

#of Playlists	1,000,000
# of tracks	66,346,428
# of unique tracks	2,262,292
# of unique albums	73,4684
# of unique artists	29,5860
# of unique titles	92,944
# of playlists with descriptions	18,760
# of unique normalized titles	17,381
Average playlist length	66.346

For our assessment we will be focusing primarily on relation of tracks. With 1,000,000 playlists and 66,346,428 tracks (of which 2,262,292 are unique), we believe we will have a sufficient dataset to evaluate what relation, if any, tracks have with each other and if track alone is sufficient for recommendations of playlist extensions.

The top tracks are HUMBLE. by Kendrick Lamar (46,574 instances; 4.66% of all playlists), One Dance by Drake (43,447 instances; 4.34% of all playlists), and Broccoli (feat. Lil Yachty) by DRAM (41,309 instances; 4.13% of all playlists). In more sophisticated analysis of the data, we could take a weighted approach like in *Efficient K-NN for Playlist Continuation* and weight these high frequency songs less than ones on the other end of the spectrum[5]. We will not be taking this approach but it may be an interesting next step to our analysis.

Only 665 songs appear have an instance rate of 10,000 or more. Even if we assume that a song can only appear on any playlist once, this means that at best, these songs only appear on 1% of all playlists. The rest of the 2,261,627 tracks will have a reoccurrence on increasingly smaller number of playlists.

A look at a histogram of playlist length indicates that 20-30 tracks is the highest concentration of tracks when we look at all of the playlists. The intent of our investigation is not to determine the optimum number of songs for an engaging playlist so we will simply use these values as an assumed benchmark for the length of playlist extension that we provide.

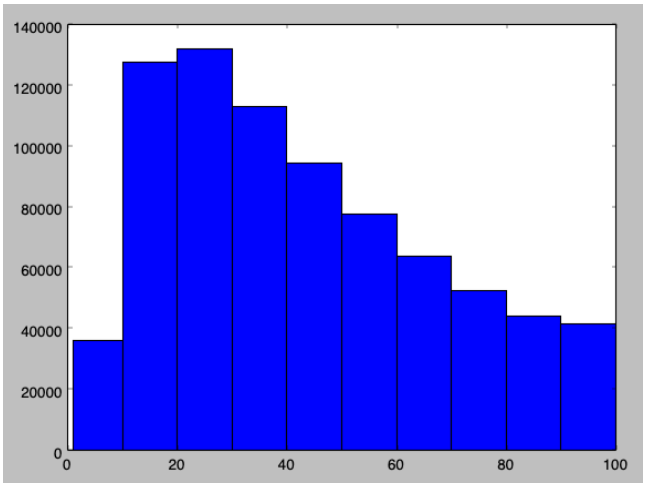

Figure 1 Histogram of Playlist Lengths

If we look a little more at the data, we see that the first quartile is at 26 tracks, the median is 49 tracks, the average is 66 tracks, and the 3rd quartile is 92 tracks. This indicates that the data is right skewed (which we could also see in the above histogram). A box plot of the full data set in terms of playlist length is below.

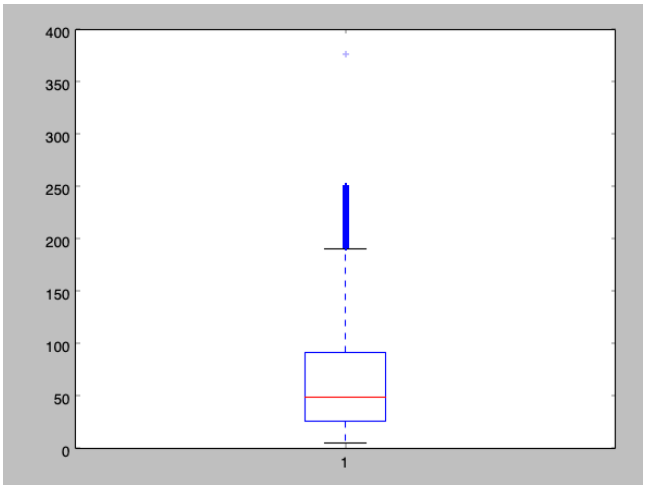

Figure 2 Box Plot of Playlist Lengths

This information is encouraging for our investigation as most playlist are of a reasonable size, determining relationships amongst songs should be at least probable even if the occurrence for any given song happens to be relatively low.

4 Main Techniques Applied

4.1 Data Quality

For our analysis we are assuming that there is no data cleaning or transformation that needs to happen prior to any data mining. We are able to assume this do to the completeness of the dataset that we are given and that this dataset was a part of a public datamining challenge. As such, it has already been cleaned, integrated, and transformed. The public information provided about this challenge claims that playlists

> "are manually filtered for playlist quality and to remove offensive content… as such, the dataset is not representative of the true distribution of playlists on the Spotify platform, and must not be interpreted as such in any research or analysis performed on the dataset"[2].

This dataset also came with an already separated evaluation set. This allowed us to use the full dataset for our analysis (where possible) instead of having to separate the data into an evaluation set and a test set.

There were a number of metrics provided by the public challenge to be able to use as a benchmark for the relative success of our playlist extensions. Most of the metrics measured features of the results that are out of scope for our analysis however. The one metric that we would use in our evaluation is the R-Precision metric which

> "is the number of retrieved relevant tracks divided by the number of known relevant tracks (i.e., the number of withheld tracks)"[2]

4.2 Initial Cluster Analysis

Our initial look of the data was in the form of a network graph. The purpose of using this type of visualization is to "assess how alike or unalike objects are in comparison to one another."[3] We were trying to get an early understanding if we would be able to take song title alone and build up clusters of likely networks that could be used to justify playlist extensions if we were given at least a single node in that network.

Unfortunately, given the size of the database, the limited hardware resources available to us, and the available software tools we were able to find for free, we were only able to do this on a small subset of the overall data. Even with this constrained view of the data we were able to glean some valuable insights.

The network diagram appeared to indicate that most songs (nodes) would have a large social network associated to them. For these songs, we would assume that we would likely have a better chance for

determining "good" fit playlist extensions given then number of neighbors and frequency for later analysis.

There also appear to be a few outliers in addition to the main cluster. Assuming that this is a trend throughout the data, and not a product of our inability to put together a cluster of the entire dataset, we can assume that there will be some songs for which playlist extensions suggestions could be limited to the playlist (or playlists) associated with that particular song. For these playlist extensions, we would expect a lower positive association with the extension given the limited information our analysis would use to provide it.

Figure 3 Full Network Diagram of First Set of Playlists

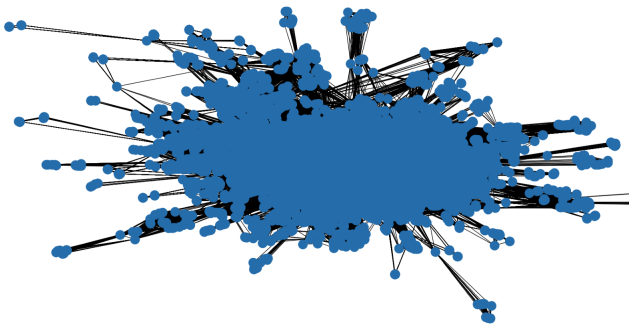

Figure 4 Zoomed In Network Diagram of the Main Cluster Above

4.3 Apriori Evaluation

We decided that our first real attempt at a playlist extension should be based on an Apriori evaluation of the data.

> "The Apriori algorithm is a seminal algorithm for mining frequent itemsets for Boolean association rules. It explores the level-wise mining Apriori property that all nonempty subsets of a frequent itemset must also be frequent. At the *k*th iteration (for $k >= 2$), it forms frequent k-itemset candidates based on the frequent $(k - 1)$ – itemsets, and scans the database once to find the complete set of frequent *k*-itemsets, L_k."[3]

We chose to start with Apriori because it is a basic frequent pattern method for data mining but can be applicable to very large data sets like the one available here through its efficiency and scalability. We felt that it would be a suitable analysis technique to use since we were only looking at the association of songs to one another. By looking at the singular quality of the playlists in this way, we could use this type of algorithm to determine support for relations of songs to one another.

This type of analysis allows us to iterate over the data, collecting songs that could be candidates for a playlist extension based on their ability to satisfy a minimum support.

Our implementation utilizes the Efficient-Apriori python package. The algorithm extracts the seed playlists from the provided challenge dataset. It then feeds each song from each seed playlist into an algorithm that scans the database for playlist matches with that song, runs an Apriori analysis and returns the top candidate. Playlists generated from seeds are then saved in csv files.

Pseudocode for our process is below:

- Process_playlist_single_seed
  - 'Main Function' → incorporate helper functions
    - Open seed CSV and extract seed playlists and playlist IDs
    - For each song in seed playlist:
      - Search 50,000 playlists in dataset for song matches
      - Run apriori market basket analysis on playlist matches
      - Generate output playlists from top candidates
        - No seed or duplicate tracks, 500 tracks total
    - If < 500 songs after going through each seed song:
      - Cut support in half and try again
    - If not enough data in first 50,000 playlists:
      - Move on to next 50,000

Figure 5 - Apriori Pseudocode Overview

There were a number of items in this code that were written specifically for this challenge. For example, to be able to have the code processed by the public challenge we needed to have the data saved in a particular format (CSV) and the playlist extension had to be exactly 500 tracks long. Per challenge criteria, there also could not be any duplicate tracks nor seed tracks in the newly generated playlist. We also had to make some decisions based off our hardware capacity which is why we initially limit the number of playlists that we search through.

In the end, we decided to run the code only on a single seed of 1000 playlists due to time constraints. As Apriori is not necessarily the most efficient algorithm due to its need to rely on multiple database scans, we found that running the code on a single seed of 1000 playlists took 10-20 hours. For this reason and we were not able to submit generated playlist to the full challenge as it required a full set of 10,000 generated playlists, one for each seed playlist in the challenge set.. The CSV with 1,000 generated playlists is posted in the results section on GitHub

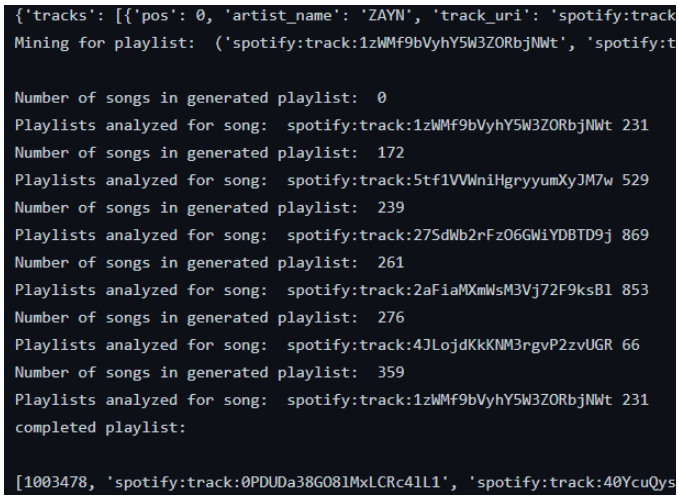

Figure 6 – Code Example Walkthrough

The figure above demonstrates terminal output for generating a single playlist given a seed playlist. It starts at the track_uri ending in JNWt and finds 231 playlist matches within the first 50,000 thousand playlists. From these playlists, it derives 172 tracks using a confidence level of .1 and adds them to the generated playlist. It then moves on to the next track. After getting through all 5 tracks in the playlist, there are only 359 out 500 tracks needed in the generated playlist. The code then returns to the first seed track ending in JNWt and cuts the support in half to .05. Although not explicitly shown, the code finds an additional new 141 tracks based on this support level and completes the playlist. The full seed playlist and outputted generated playlist can be seen in the text file in the results section on GitHub.

4.4 K-Nearest Neighbors

We had intended to utilize K-Nearest Neighbors as a part of our endeavor. We understood that,

> "Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it. The training tuples are described by *n* attributes. Each tuple represents a point in *n*-dimensional pattern space. When given an unknown tuple, a *k*-nearest-neighbor classifier searches the pattern space for the k training tuples that are closest to the unknown tuple. These *k* training tuples are the *k* "nearest neighbors" of the unknown tuple."

This seemed to be an appropriate means to evaluate the date. We could take a portion of it and create a data set, then look for candidates from that training set to extend a playlist prompt by finding its *k* nearest-neighbors. Unfortunately, with our implementation of looking at tracks, we could not determine how to establish a Euclidean distance amongst qualitative information. We decided to find another approach to solving the question.

4.5 Natural Language Processing

The approach we shifted to after K-Nearest Neighbors was using Natural Language Processing (NLP). Although this was not a topic specifically covered in this course, we felt that it embodied the spirit of data mining through text mining and applying it as one possible way to solve this problem. More sophisticated

approaches, like the one employed by Ludwig et al.,[4] could use this as another piece to the overall data mining process. As a reference point,

> "Natural language processing (NLP) refers to the branch of computer science – and more specifically, the branch of artificial intelligence or AI – concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.
> NLP combines computational linguistics – rule-based modeling of human language – with statistical, machine learning, and deep learning models."[8]

We were further encouraged by a passage in the textbook for this course, which read,

> "The power of data mining can be substantially enhanced by integrating new methods from multiple disciplines. For example, to mine data with natural language text, it makes sense to fuse data mining methods with methods of information retrieval and natural language processing. As another example, consider the mining of software bugs in large programs. This form of mining, known as bug mining, benefits from the incorporation of software engineering knowledge into the data mining process."[9]

Considering this viewpoint, we saw NLP as a low-cost complement to the Apriori algorithm. Because this approach would be less resource-intensive than Apriori, the use of alternative NLP methods might serve to support and corroborate the results of the more sophisticated techniques. Furthermore, because playlist creators have the freedom to describe their playlists in "fun" and unique ways, NLP is a good candidate for mining playlist titles.

Using the Natural Language Toolkit module in Python, our implementation closely followed the approach designed by Ludwig. Using some prompt, we perform a "fuzzy" search against the playlist names, which are tokenized (separated into constituent elements), stemmed using the Porter algorithm (removing the suffix to arrive at the root of the word), and normalized to lower case. The search uses Levenshtein distance (a measure of the similarity between two strings) of 1 to get close, but not necessarily exact, matches. After identifying candidate playlists, we gather all of the songs contained in the playlists and return the top 10 most frequent songs. As an example, using "run" as a prompt, 7,756 candidate playlists were discovered, with a total of 436,925 songs. Sanity checks show that these are "upbeat" songs that one could conceivably engage in sporting activities to.

We are further encouraged by these results and see this method being adjusted to process song titles instead of playlist titles. In this way, the NLP approach could be used to support Apriori or other classification methods.

5 Key Results

There were lots of results that we found through this process. The main takeaway we had is that there needs to be an understanding of where the value for incremental accuracy starts to fall off into diminishing returns. We found that there are a number of ways to solve this type of problem with varying degrees of accuracy. Often the ones with less accuracy run quicker. For example, our Apriori analysis may have been more statistically accurate than our NLP analysis, but the difference in runtime amounted to dozens of hours versus minutes for the entire dataset. The run time, therefore, needs to be part of the assessment of which methods to use. This was even evident in some of the research we saw on this subject. Other groups had commented that they had returned results that were very close in score to groups that had scored higher, but with algorithms that ran much faster and were capable of running on standard equipment.

The first interesting result that we came across was with playlist lengths. We noticed that the lengths of playlist was right skewed. That the average was around 66 tracks, but when you looked at a histogram of the data, we saw that the peak of playlist length resided

more around 20-40. This makes intuitive sense when you think about it. If an average song is 3 minutes and 30 seconds[7], that is between 70 minutes and 210 minutes in total. That is a substantial amount of time for most folks to sit around listening to music, especially in today's world which seeming seems to put value on short forms of entertainment.

This finding on playlists is interesting, especially given the challenge to extend the playlist to 500 tracks (1,750 minutes or close to 30 hours). This is certainly longer than most people would spend listening to a playlist. It does however provide a good place to practice applications of data mining. As mentioned earlier in this paper though, another interesting extension of this premise would be to see at what length is the optimum playlist?

A second result that we found to be interesting was in the network relation of songs in our dataset. There appeared to be a vast majority of songs that were connected to each other through the various playlists made available to us. An assumption of this is that these songs are likely songs that are more mainstream in their availability and therefore more likely to show up purely from a frequency perspective. There are other songs that seem to be only related to the other songs in their particular playlist or very few other playlists with equally unique songs. To be clear, due to limitations of the hardware available and the sophistication of our algorithm and charting tool, we were limited to only one file worth of playlists to make the following determination.

A naïve conclusion that can be drawn from this is that linking popular songs to other popular songs should give at least some reasonable success in a playlist extension. The converse would likely also be true, that linking the very infrequent songs to the other songs that they are linked to already would generate a reasonable (if not small) playlist extension. How successful in terms of user acceptance may be another question, however. This type of conclusion seems to have some legs to it as it was, at least in part, used for weighting of algorithms in one of the other groups working on this challenge in the past[5].

Another key finding was the plausibility of using Apriori as the basis for a recommendation system. Although the run time was significantly longer, we were still able to reliably generate playlists for 1000 different seed playlists from the challenge dataset.

If we look at how Spotify operates as an end user, we can see that it generates playlists on different schedules. For example, it will generate playlists like the Release Radar and Discover Weekly which are updated on a weekly basis and also Daily Mixes which are updated daily. This demonstrates the usefulness of a longer analysis such as Apriori to generate playlists that are perhaps more interesting.

If Apriori was slow, inefficient and resource intensive, our NLP solution was blazingly fast. On modest hardware (middle-of-the-road laptop built in 2018), our NLP algorithm returns a result in just over five minutes. There is tremendous value in methods that achieve a significant portion of the results of another approach in a fraction of the time. Of course, NLP isn't meant to replace proper classification or clustering data mining techniques. Rather, it serves to complement other approaches, either as a secondary measure or simply as another approach that sees the data from a new angle.

In choosing this approach, we considered the way people interact with playlists. Increasingly, people turn to playlists that other people created to discover new music. These people might use the playlist title to select a playlist, so NLP feels like a natural fit for music discovery. In fact, Spotify could generate lists of playlists to suit listener mood, solely based on playlist titles.

We would be remiss if we didn't highlight some of the negative aspects of NLP. For one, because users have autonomy over playlist titles, it is an inconsistent and unreliable source. As an example, descriptive words that lend themselves to data mining and association like "run", "dance", "sadness" or "country"

lead to satisfying results, while "My Awesome Playlist" has no discernible relationship to the songs contained in the playlist. As such, these playlists are ignored by our algorithm. Additionally, the rise of the emoji as an element of expression confounds our NLP approach; although this roadblock could potentially be mitigated by converting emoji into their representative meanings in text form.

6 Applications

There are any number of applications for this type of analysis from extending music playlists (as was the direct application here), providing suggestions for new content on video streaming applications, suggesting new connections on social media applications, or providing frequently bought together type suggestions for virtual shopping. The above applications are not meant to provide an exhaustive list of potential applications.

The extension into these other areas with this type of analysis is possible, in part, due to the simplicity of assumptions. We were only looking at song title in relation to other song titles in (or not in) a playlist. We did not factor in other considerations like genre, playlist name, follows, etc. This does perhaps leave room for more in depth analysis of this data but it also leaves this analysis much more open to other applications. It requires virtually no knowledge of the space being evaluated for use. This is likely most helpful for spaces like music, movies, and any other subject space where naming of the product (song), manufacturer (artist), or even category (genre) can be arbitrary at best.

Extension of this type of analysis to providing suggestions of new content on video streaming applications is relatively straight forward. It would make the same base assumption that the name of the video, category (or categories), and other metadata type information would be relatively arbitrary. That simply by looking at frequent patterns of viewing/saving to "liked" lists, one could recommend other videos with reasonable accuracy. This type of recommendation system would likely be valuable as it is assumed that value for these platforms is likely driven by the perceived content quality rather than quantity i.e. users ability to find content that they really enjoy regardless of the amount of time they spend on the platform is more valuable than the amount of time they spend on the platform.

Social media applications could (and presumably do to some extent) use this type of analysis to suggest growth of connections on the platform. This type of application would likely require additional analysis and information to be truly valuable. It can be assumed that people who have a lot of common connections may become aware of each other naturally and the potential to grow value to the users in this way would likely be limited. This type of application would also have to assume that social media applications value the growth of connections as a primary driver of the business model. Today, this is simply not the case for a vast majority of these applications (at least the most used applications) though. Their business models are largely driven by selling of advertising space which would have different intrinsic incentives. It is possible however that the social media application could use the information to sell advertising. By being able to show that folks who typically talk about or show products 'x', 'y', and 'z' on their associated spaces, that they are also likely to talk about and show 'w,' would allow them to have a better case for selling 'w' advertising to folks with the frequent item set of 'x', 'y', and 'z.'

Virtual shopping applications would be another reasonable application for this type of analysis. It is similar to a Market Basket analysis in that it is a frequent pattern data mining method that would allow the user to see patterns of purchase amongst users just by looking at the item e.g. the example of noticing that beer and diapers are a frequent item set. This type of use would allow shopping applications to provide recommendations to users on how to fill up their remaining shopping cart. Successful implementation of this could likely yield higher revenues for the application (through getting customers to add things to

their baskets that they perhaps would not have without the suggestion) and higher engagement with the application in general (if this could be done well enough to save the user time from having to search and add all of the items themselves). Both of these benefits could be very beneficial for the application. The first would allow essentially a virtual version of impulse buys by getting the customer to consider other items of interest. The second could improve the likelihood that the user would prefer the shopping application to other applications. It can easily be imagined that there would be great value to users shopping for groceries, if they only had to add a few items to their cart, and the rest would be automatically added/suggested for them. This could save the user real time and make an experience that most folks do not particularly like, tolerable.

REFERENCES

[1] DiMA. Digital Medial Association: Annual Music Report; **A MIDIA Research Report**. (March 2018). Retrieved October 16, 2022 from https://dima.org/wp-content/uploads/2018/04/DiMA-Streaming-Forward-Report.pdf

[2] AIcrowd. Spotify Million Playlist Dataset Challenge: A dataset and open-ended challenge for music recommendation research. Retrieved October 16, 2022 from https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge

[3] Jiawei Han, Micheline Kamber, Jian Pei "Data Mining: Concepts and Techniques", 3rd Edition, Morgan Kaufmann, 2011. '

[4] Ludewig, M., Kamehkhosh, I., Landia, N., & Jannach, D. (2018). Effective nearest-neighbor music recommendations. In *Proceedings of the ACM Recommender Systems Challenge 2018* (pp. 1-6).]

[5] Kelen, D. M., Berecz, D., Béres, F., & Benczúr, A. A. (2018). Efficient K-NN for playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018* (pp. 1-4).

[6] ACM Digital Library. Research Article. *Efficient K-NN for Playlist Continuation*. Retrieved December 4, 2022 from https://dl.acm.org/doi/abs/10.1145/3267471.3267477

[7] Digital Music News. What's the Average Length of a Song? It's Shorter Than You Think. Retrieved December 5, 2022 from https://www.digitalmusicnews.com/2019/01/18/streaming-music-shorter-songs-study/#:~:text=According%20to%20a%20new%20report,seconds%20%E2%80%94%20and%20are%20steadily%20shrinking.

[8] IBM. IBM Cloud Learn Hub. *Natural Language Processing (NLP)*. Retrieved December 6, 2022 from https://www.ibm.com/cloud/learn/natural-language-processing

[9] Han, Kamber, Pei. DATA MINING: Concepts and Techniques, Third Edition. Waltham, MA. Morgan Kaufmann Publishers. 2012.