Music Recommender
A Music Recommendation System with Playlists

Austin Way, Bohan Liu, Christian Rodriguez Ochoa,
Kevin Peng, Kin Hei Wong, Nathan Wong, Sahana Deo,
Sean Rafael Hingco, William Shih, Zheng Yuan Chen

1 Introduction

Music coexists with human history and continuously evolves thus defining generations, ethnic cultures, and personal identity. Some of the most popular music genres include: rock, pop, hip-hop, country, RB, classical, and electronic. Taking into account existing music and music that is released every day, suggests that there are a countless amount of songs. While it's easy to get overwhelmed with such a large selection, people typically have a "go-to" list of their favorite songs. Sometimes this playlist changes depending on the occasion. For instance, a person may listen to classical music while studying and hip-hop while going out with friends. However, listening to the same songs repeatedly may become dull or even irritating.

Discovering new music has modernized thanks to machine learning techniques that arose due to the Digital Age. Traditionally, a person would find new music through word of mouth or what is being played on the radio. They may have also gotten personal recommendations by someone at a music shop or by being exposed to different cultures. While these methods are still practiced, there are now more advanced tools used to recommend songs. One of these tools is a machine learning technique known as clustering. In the context of music recommendation, clustering takes songs that a person likes and connects it to another song based on song attributes given by the dataset. With this network of connections, someone is able to find new songs to listen to based on songs they already enjoy. While this technique is widely implemented to find possible links between songs, different approaches were considered for this project.

In this report, current methods and techniques used to recommend songs were investigated. Association rule mining demonstrated to be the most practical approach to create the web application for this course: "Music Recommender"; however, its performance could be be improved with a larger dataset. Using this framework in conjunction with Frequent Pattern Growth (FP-Growth), established probability rules for all possible song selection combinations in the dataset: Spotify Million Playlist Dataset. This way, when a list of songs are entered into the Music Recommender, the probability of potential liked songs will be given. Then, the probabilities are used to return a list of the songs with the highest probabilities. Applications of this model can be used in a variety of scenarios. From providing music enthusiasts with personalized playlists to inspiring artists in pursuing newer genres, a user-friendly music recommender system has vast potential in its accessibility and popularity across all target audiences.

2 Literature Review

2.1 Overview

Within the realm of Machine Learning, two methods are primarily used for recommendation systems; K-Nearest Neighbours and Association Rule Discovery. There are additional, unlisted recommendation methods that were considered [1]. The framework for this project uses Association Rule Discovery.

2.2 Association Rule Mining

An association rule connotes a relationship between one set of items a to another set of items b, such that if the set of items a is chosen, there is a defined chance that the set of items b will also be chosen. To illustrate the idea of an association rule, an arbitrary context of a supermarket setting is used.

Analyzing customer purchases at a supermarket can reveal valuable information that may increase profits. For instance, it is found that when customers bought eggs and ham, they were very likely to also buy beer. The supermarket may apply this data to encourage the purchase of all three items by using a sale. This association rule is denoted as:

$$\{eggs, ham\} => \{beer\}$$

The left set of the association rule is the antecedent, and the right side is the consequent. Association rules, such as the one listed above, are obtained by using an algorithm that extracts a rule for each possible combination of variables. In the arbitrary example, a variable is a purchased item and a bucket is the entire set of items purchased together. Support is the default popularity of a variable. This corresponds with the proportion of the number of times eggs and ham are purchased together over the total transactions. Correctness of the rule or confidence is the likelihood that a variable will be chosen if a separate variable is already chosen. this corresponds with the proportion of the number of times eggs, ham, and beer were chosen together over the total number of times eggs and ham were purchased together [2]. For rules to be useful, a high level of Support and Confidence is to be preferred. If the aforementioned rule had the support of 90% and 50% confidence, then 90% of transactions have bought eggs and ham together, and 50% bought beer with those two.

Association Rules are generated primarily through two steps. Firstly, itemsets are found and considered "frequent" if they have a support $S1$ above minimum support multiplied by the minimum confidence CS. Once all frequent itemsets are found, for each of these sets I, remove an arbitrary item J to produce a potential consequent set. If set I' comprised of $I - J$ has a support $S2$ above the minimum support, and $S1/S2$ is larger than minimum confidence C, then a suitable association rule has been found. [5]

The largest problem with generating Association Rules is the generation of frequent item sets, which requires large degrees of space and computational power. The generation of all possible pairs within the set of all items for counting would take $O(N^2)$, not to mention counting all triplets, quadruplets, etc, such that writing a naive algorithm to generate association rules is unfeasible for large datasets [5].

2.2.1 Apriori Algorithm

The Apriori Algorithm, created by Rakesh Agrawal and Ramakrishan Sikrant [3], is the most common approach for generating association rules. While there are numerous small-scale optimizations in terms of data structures to reduce bottlenecks, the primary optimization is the exploitation of the feature of monotonicity; if a set of items appears at least s times, so does every subset of the set and its contrapositive: for Association set pairs, if item i does not appear in S items, then no pair including i can appear in S items. Largely summarized and simplified, the Apriori Algorithm works in passes, generating frequent item sets of k size in each pass. For each pass, items that are not considered frequent are disregarded as candidates for checking for the $k+1$ pass, thus drastically reducing the number of candidates to be checked for frequency. Despite these optimizations however, Apriori still ultimately needs to generate and compare a large amount of data, making it notoriously slow for larger datasets. [6]

2.2.2 Frequent Pattern Growth

The Frequent Patter Growth (FP-Growth) is an alternative approach to rule mining that builds a compressed Frequent Pattern Tree data structure out of the transactions, then mines that tree via a divide and conquer method to get frequent item sets. The specifics of this algorithm are beyond the scope of this project as the outputs are the same, but the main advantage of using a FP-Tree is such that no item set generation is needed and the usage of a tree data structure thus allows FP-Growth to mine at an order of magnitude faster than the Apriori Algorithm. [4]

2.3 Further Research

During the initial stages of the project research, two music recommendation system studies were reviewed to understand how to create an effective recommender. In the first study, researchers Hung-Chen Chen and Arbee L.P. Chen propose several methods that are considered standard methods. The group understood that collaborative filtering and content-based filtering are established ways to filter data and look for patterns that can help in recommendations. Collaborative filtering refers to the process of evaluating similarities between distinct user profiles. In the case of the music recommendation system, these profiles would be users who enter in their own data (artist or song title as an example). Collaborative filtering would ensure that similar user profiles are placed/grouped together. These distinct groups would then receive recommendations that pertain to their cluster. Content-based filtering, however, deviates from comparisons between user profiles. It still views user profile data, but it filters for similarities between data items and user profiles [7]. Both methods can also be used together. Collaborative filtering can dive into deeper analysis of similar features/interest between relevant users [7]. The second paper of interest introduced provided an introduction to context-aware music recommendation system (CA-MRS) which incorporates fuzzy Bayesian networks. It proposes that context is an important condition to study, as user preferences on music can change over time [8]. By conducting tests on user preferences for music attributes and other context features (mood, genre), the researchers suggest that this is also a viable technique which can be used alongside the aforementioned approaches.

3 Dataset Description

3.1 Overview

The dataset used for the Music Recommender project, is the Million Playlist Dataset. This dataset is in the form of multiple JSONs which contain a lot of data. The data includes the name of the playlist, whether the playlist was made collaboratively, when the playlist was last modified, how many unique albums are included in the playlist, how many tracks are included in the playlist, the number of followers of the playlist, the number of times the playlist was edited, the total duration of songs for the playlist, the number of unique artists in the playlist, and the data for the tracks themselves. The tracks themselves have additional data. This data includes the position of a specific track in the playlist, the artist name of the track, the track's uniform resource indicator (URI) for Spotify, the artist's URI for Spotify, the track's name, the album's URI for Spotify, the duration of the song, and the name of the album for the track.

3.2 Conversion

A portion of the playlists were extracted from the Million Playlist Dataset because the original size of the dataset was too large to process. Next, the JSON format was converted into csv format with only the artist name and track name as data properties. The resulting csv file has each line as a single playlist with each element being a track that includes the artist name and track name. This csv file was used to create associations using Association Rule Mining.

4 Solution and Results

4.1 Solution Overview

The dataset chosen was Spotify's Million Playlist Dataset for lack of better dataset. Because there was no individual rating metric for each of the songs in every playlist, the ML Team felt that attempting to use K-Nearest Neighbours would not be the best approach. Most approaches to collaborative filtering utilize some sort of rating metric, and thus attempting to fit in purely binary data, on presumably hamming distance is going to end up looking weird. K-Means Clustering was also not an option given the purely binary data would not fit well either.

Association Rule Discovery instead perfectly fit within the constraints of the dataset, since we only need to care about the occurrences rather than the ratings of songs with respect to the entire dataset. Mlxtend Library was used to implement the approach.

4.2 Initial Approach

Given the large size of the Million Playlist Dataset and the expensive costs of Association Rule Discovery, three portions of size 500, 1000, 5000, and 10,000 rows were taken for possible implementation. When encoded into a One-Hot Encoding in which columns were songs and rows were playlists, it turned out that the number of columns greatly exceeded the number of rows, with 500 rows containing 20,000 songs as shown in Figure 1. Additionally, it was observed that 5000 rows contained 70,000 songs. This makes sense, as since each playlist can contain a multitude of songs. However since there is ultimately a limited number of possible songs, the ratio between playlists and songs would likely even out as the number of playlists increased. Unfortunately, due to the large number of columns, just reading the 5000 sample into a dataframe took around 59% of our 16gb of RAM, making larger sizes untenable since extra space was also needed for the frequent generation.

Figure 1: Sample of 500 Playlists

	missyelliott_losecontrol(feat.ciara&fatmanscoop)	britneyspears_toxic	beyoncé_crazyinlove	jus
0	1	1	1	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	
...	...	...	...	
495	0	0	0	
496	0	0	0	
497	0	0	0	
498	0	0	0	
499	0	0	1	

500 rows × 20574 columns

Initially, the Apriori Algorithm was used to find frequent item sets on the 500 sample shown in Figure 2. However, because the number of songs greatly exceeded the number of

rows, relatively high support rates such as 0.6 or even 0.1 yielded very little frequent item sets. As such, a confidence rate of 0.01 was set instead, in that a item set that occurred at least 5 times was considered frequent. This yielded a few thousand frequent item sets.

Figure 2: 500 Sample Association Rules

	antecedents	consequents	antecedent support	consequent support
0	(britneyspears_toxic)	(boyslikegirls_thegreatescape)	0.012	0.020
1	(justintimberlake_rockyourbody)	(beyoncé_crazyinlove)	0.012	0.032
2	(solidstar_mybody)	(beyoncé_crazyinlove)	0.014	0.032
3	(justintimberlake_rockyourbody)	(usher_yeah!)	0.012	0.034
4	(lisaloeb&ninestories_stay)	(usher_yeah!)	0.010	0.034
...	...	...	...	...
1929	(oscarisaac_green, travisatreo_youareinlove)	(lukebryan_suntancity, lukebryan_nightone, kan...	0.012	0.012
1930	(lukebryan_nightone, travisatreo_youareinlove)	(lukebryan_suntancity, kanyewest_gorgeous, osc...	0.014	0.010
1931	(lukebryan_suntancity, mileycyrus_fu)	(lukebryan_nightone, kanyewest_gorgeous, oscar...	0.014	0.010
1932	(lukebryan_suntancity, oscarisaac_green)	(mileycyrus_fu, lukebryan_nightone, kanyewest_...	0.012	0.010
1933	(lukebryan_nightone, oscarisaac_green)	(mileycyrus_fu, lukebryan_suntancity, kanyewes...	0.014	0.010

1934 rows × 9 columns

In generating association rules from these frequent item sets, once again, a high confidence value of 0.7 produced only around a dozen rules. As such, a confidence value of 0.5 was set to make around $\tilde{1}200$ rules.

While these rules were usable, the low support and confidence values meant that they were not very good rules. The small sample rate also meant that these rules only encompassed a small portion of all the given 22,000 songs. Attempts were made then to try to utilise the 1000 and 5000 sample. At this point however, the very large number of songs meant that computational resources were stretched beyond limit. The problem was that in setting high support rates, these sample yielded rules of little or similar size to the 500 sample. When setting a an even lower confidence value, Python simply ran out of memory to store all the item sets for each pass in Apriori. The 5000 sample for example, required around 1.7 terabytes of memory to run.

4.3 Frequent Pattern Growth Approach

Given the problem of running into space limitations rather than processing limitations, utilizing Apriori would be untenable. With further research, it was concluded that using the Frequent Patter Growth Algorithm (FP-Growth) would be more logical.

Unlike the Apriori Algorithm which generates and checks candidates for frequency in multiple passes, FP-Growth utilizes a "Frequent Pattern Tree" data structure to search for frequent item sets with a divide and conquer strategy. This approach greatly reduces computational time, but more importantly does not require much extra space compared to the Apriori Algorithm.

With the FP-Growth Approach, frequent item sets of support 0.002 on the 5000 sample were successfully generated as shown in Figure 3. This is a higher absolute (10) support than the 500 sample. Approximately 14,000 item sets were generated.

Figure 3: 5000 Sample

	missyelliott_losecontrol(feat.ciara&fatmanscoop)	britneyspears_toxic	beyoncé_crazyinlove
0	1	1	1
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
...	...	...	...
4995	0	0	0
4996	0	0	0
4997	0	0	0
4998	0	0	0
4999	0	0	0

5000 rows × 72834 columns

From this, a minimum confidence of 0.7 yielded approximately 200,000 association rules as shown in Figure 4. Thus, a large number of rules were efficiently extracted comparable to the number of songs in the Million Playlist Dataset.

Figure 4: 5000 Rules

	antecedents	consequents	antecedent support	consequent support
0	(beyoncé_crazyinlove, nellyfurtado_promiscuous)	(usher_yeah!)	0.0026	0.0150
1	(iyaz_replay, souljaboy_kissmethruthephone)	(jasonderulo_whatchasay)	0.0028	0.0076
2	(jessemccartney_leavin')	(jessemccartney_beautifulsoul)	0.0026	0.0068
3	(bonjovi_livin'onaprayer, survivor_eyeofthetiger)	(journey_don'tstopbelievin')	0.0036	0.0188
4	(queen_anotheronebitesthedust-remastered2011, ...	(journey_don'tstopbelievin')	0.0028	0.0188
...	...	...	...	...
228988	(brantleygilbert_smalltownthrowdown)	(brantleygilbert_bottomsup)	0.0028	0.0052
228989	(samhunt_raisedonit)	(samhunt_leavethenighton)	0.0030	0.0104
228990	(j.cole_wetdreamz, j.cole_crookedsmile)	(j.cole_norolemodelz)	0.0024	0.0218
228991	(j.cole_norolemodelz, j.cole_crookedsmile)	(j.cole_wetdreamz)	0.0024	0.0128
228994	(russelldickerson_bluetacoma)	(samhunt_bodylikeabackroad)	0.0024	0.0150

228953 rows × 9 columns

5 User Interface / Web Application

To create the user interface for Music Recommender, a combination of plain HTML, CSS, and JavaScript was used resulting in a standard web application. When the user opens the application, they are presented with a search bar, a list of random songs, and an empty playlist; they then are able to add to the playlist via a button next to each song. Using the search bar, the user can also search through the data set by song title. When they are done adding songs, they can click the "Recommend Songs" button and the application displays a list of recommended songs.

Since training the model returned a static list of association rules, there were simply added into the web page via JavaScript without usage of any outside plugins. This also means that the performance of the app increased significantly compared to other methods. This is because the only computation required for song recommendation is a search through the list of association rules (for both antecedents and consequents). [9]

6 Conclusion and Discussion

From this exploration, it has been quite clear that relying upon association rules solely as a recommendation system is not a very good approach for small scale projects. Association rule mining does not scale very well with large data sets, yet large data sets are needed for useful recommendation systems. In the scenario whereby a million playlists are actually analysed, the number of playlists would be comparable to the number of songs, allowing for higher levels of support, but analysing a dataset of that size is going to require gargantuan levels of computational resources. Moreover, because of how high support and high confidence correlate to having a high proportion of playlists having similar tastes, the number of good rules that can be generated is always going to be covering a relative minority of users. For a small yet diverse dataset, it could end in having no recommendations occur at all!

Furthermore, within the usage of generated rules, checking if around 1000 rules might apply to an input might be easy, but when the number of rules is in the hundreds of thousands, or a even millions, continuously checking and applying for occurring data sets is not going to scale very well.

Recommendation Systems do not need to solely rely upon Machine Learning models, and content based systems such as similar genres or artists can be also used to supplement these models when they are insufficient alone.

7 References

References

[1] Pavel Kordik. *Machine Learning for Recommender systems.* https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed

[2] Rakesh Agrawal, Tomasz Imieliński, Arun Swami. *Mining association rules between sets of items in large databases.* IBM Almaden Research Center, San Jose, 1993.

[3] Rakesh Agrawal, Ramakrishnan Srikant. *Fast Algorithms for Mining Association Rules.* IBM Almaden Research Center, San Jose, 1994.

[4] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. *Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach.* Simon Fraser University, Canada, 2000

[5] Jeffrey Ullman *Lecture 20 — Frequent Itemsets — Mining of Massive Datasets.* Stanford University
https://www.youtube.com/watch?v=O9QnC5WJJ90

[6] Jeffrey Ullman *Lecture 21 — A Priori Algorithm — Mining of Massive Datasets.* Stanford University
https://www.youtube.com/watch?v=tY1JE6XFjCYt=559s

[7] Hung-Chen Chen, Arbee L.P. Chen *A music recommendation system based on music data grouping and user interests.* National Tsing Hua University
http://make.cs.nthu.edu.tw/makeWeb/DBPaper/pdf/CC-01.pdf

[8] Han-Saem Park, Ji-Oh Yoo, Sung-Bae Cho *A Context-Aware Music Recommendation System Using Fuzzy Bayesian Networks with Utility Theory.* Yonsei University
https://www.mi.fu-berlin.de/en/inf/groups/hcc/teaching/Past-Terms/Winter-Term-$2016_1 7/Paper/10_c ontext_F uzzy.pdf$

[9] ECS 171 Group 14 Source Code, GitHub Repository
https://www.github.com/kevinp2000/ECS171-music-recommender