Music Recommendation System

Himanshu Gupta
himanshu.gupta@colorado.edu
CSCI 5502, University of Colorado Boulder
Boulder, Colorado

Jaeyoung Oh
jaeyoung.oh@colorado.edu
CSCI 5502, University of Colorado Boulder
Boulder, Colorado

Youngjin Ko
youngjin.ko@colorado.edu
CSCI 5502, University of Colorado Boulder
Boulder, Colorado

Yutaka Urakami
yutaka.urakami@colorado.edu
CSCI 5502, University of Colorado Boulder
Boulder, Colorado

Figure 1: Obtained from https://www.dezyre.com/project-use-case/music-recommendation-challenge

ABSTRACT

Nowadays, there are various types of platforms (video, music, shopping, etc) that provide recommendation services for users. These recommendation systems not only help novice users who just started on the new platform but also enhance the experience for current users who have been using the platform for quite some time by analyzing past searching history and suggesting content that users are more likely to be interested in. In this project, we aim to make a recommendation system from 'Million Song Data' using tools that we learned in the Data-mining course with Prof. Qin Lv.

KEYWORDS

Data Mining, Data sets, Music Recommendation, Apriori, Collaborative Filtering

1 INTRODUCTION

For decades, thanks to the propagation of the network and connectivity, and popularization of smartphones, music platform subscribers have grown exponentially as can be seen in figure 8. Nowadays, the online music market has became a huge industry. Accordingly, the recommendation system also has become an essential part of it. By using the recommendation system, a customer can come across content that are tailored to his needs and can make a decision more easily. The better quality of recommendation will improve the user experience and the consumers will feel more satisfied with the service and this can lead to them subscribe for longer duration. Therefore, lots of media companies invest in developing excellent recommendation engine.

From 2006 to 2009, Netflix-sponsored competition offered a grand prize of 1,000,000 to the winning team. They provided a data set of over 100 million movie ratings, and the winning criteria was that the new recommendations were at least 10 percent more accurate than the predictions from their existing recommendation system. [11]

There has already been a lot of work done in the field of recommendation systems. Nevertheless, the ground theory of recommendation system is Data-mining. Therefore, we will recursively build the frame of recommendation service with Data-mining tools and apply to a subset of the massive data set 'Million Song Data'.

Through this project, we can figure out how accurate our recommendation system is, comparing with modern commercial recommendation systems and finding out the limitation of our methods. The huge scale of data, ability to mine data from it and solving a real life problem makes it interesting to our group. One of the major challenges is also the huge data set and developing anything using it is going to be computationally expensive and thus difficult. As the

size of 'Million Song Data' is substantial, expected processing time is variable and depends on external factors such as the Capacity of RAM, processing size(32bits, 64bits). Also, people have been doing extensive research in this field for a long time now. Devising a system that can improve the existing state of the art recommendation techniques, and comparing results with other techniques, and understanding why one technique works better than the other is not straightforward and hopefully we will try to address that by the end of this project.

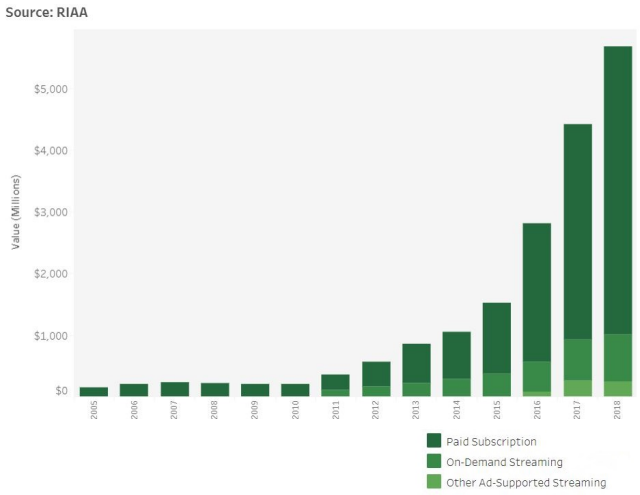

Figure 2: 2005 - 2018 U.S. Music streaming service growth (https://www.riaa.com/u-s-sales-database/).

2 RELATED WORK

Music is prevalent. However, the collection of music that music streaming services have these days is so huge that it is not easy to find and choose the music that a user wishes to listen to. Searching anything on such huge mount of data gives a lot of results, and it is hard to find the right one. This makes a recommendation system essential for finding out the type of music each user will prefer. Recommendation can be considered as an alternative to searching. Every music company provides a personalized playlist with its algorithm[3, 9]. The most frequently used algorithms are mentioned below briefly: [8, 10].

- `Collaborative filtering (CF)` :Collaborate filtering depends on analyzing and gathering a lot of information about user behavior, activities, or preferences, and predicting what users will like based on similarities with other users. One important advantage of this approach is that it does not rely on understanding the meaning of content. Hence, you can recommend multiple items like movies without accurately understanding the item itself. Apple relies on this algorithm to recommend song in iTunes.
- `Context-based filtering` :This algorithm relies on a description of the item and a profile of the user. In the Content-based recommendation system, keywords are used to describe an item, and user profiles define the type that the user likes. This algorithm recommends similar items that users used to like or what they are currently seeing. Specifically, various candidate items are compared with items rated by the user, and the system recommended best-matching items. Internet radio company Pandora relies on this system to make personalized internet radio station.
- `Hybrid approach` :The hybrid approach is a mixture of CF and content-based filtering, which may be more effective in some situations. This method can overcome the common problems in recommendation systems such as cold start and sparsity problems.

There are several scholarly works of literature for music recommendation. They provide novel approaches to improve recommendation accuracy. In the paper[1], they are trying to build a personalized internet radio stream with a new probabilistic collaborative filtering model. They used the playlist of thousands of internet radio stations, which include millions of plays and hundreds of thousands of tracks and artists. They also did cross-source validation. The model trained on the internet radio data was transferred to the Yahoo behavior prediction successfully. Another beneficial approach is matrix factorization.

The paper[8] relies on the method. They improve Yahoo music recommendation accuracy. They suggest the matrix factorization method with temporal analysis of user ratings, and item popularity trends with Yahoo music data set, which include a million users, 600 thousand musical items, and 250 million ratings for a decade. This approach achieves the integration of taxonomy information of items and different music ratings.

The paper[4] suggests the Latent Markov Embedding approach to generate an automatic playlist. The data used in the article are radio playlist from yes.com, which has hundreds of stations in the U.S. They optimized the problem with the regulated maximum-likelihood embedding of Markov chains and solved efficiently.

The last paper[6] shows a context-aware music recommendation system. The data is human-compiled music playlists, which include 7,051 playlists and 21,783 songs. They provide a topic modeling with the latent topics generated from the most frequent tagged songs. The method has a more accurate recommendation result than a conventional system that relies on CF or content-based filtering.

3 DATASET(S)

We plan to use two existing datasets: Kaggle's competition, Million Song Dataset Challenge[7] and Million Song Dataset. [12]

The data set in the Kaggle competition comprises of : 1) the 386,213-song IDs; 2) the 110,000-user IDs; 3) the listening histories of the 110,000 users; 4) the mapping from song IDs to the track IDs.

The Million Song Dataset uses the same song and track IDs and has detailed information about those tracks. It includes 1) the mapping from the song IDs to the song title and artist name; 2) the mapping from the track IDs to the track descriptions (artist name, danceability, duration, energy measure, key, loudness, year, tempo, and so on); 3) the mapping from the artist IDs to his/her locations (latitude, longitude, city, and state)

4 PROPOSED WORK

In this section we discuss about the strategies that we wish to implement and the motivation behind them.

4.1 Apriori Algorithm

Apriori algorithm [13] is used for mining frequent item sets and determining strong association rules over relational databases. It proceeds by identifying all the frequent 1-item sets in the database. It then generates all the possible 2-item set candidates by combining these 1-item sets and choose those item sets that appear sufficiently often in the database. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time. The algorithm terminates when no further successful extensions are possible. The frequent item sets determined by Apriori can be used to determine association rules which can be used to generate recommendations or determine general trends in the database. Amazon uses this algorithm to recommend users similar items for purchasing and it has worked really well for them. The same Apriori algorithm can also be used to determine frequent song-sets and songs from the same set can be used for recommendations to all the users. We use the algorithm to find strong association rules. Our dataset provides us the listening history of users in this format- user ID, song ID, number of listenings.

We do data pre-processing and convert this into the desired format, which is

UserID - song ID$_1, songID_2, \ldots songID_N$

for all the users and then apply Apriori to it.

This approach doesn't take into consideration the number of times a particular user listened to any song. We plan to use this information in our algorithm for better recommendations.

One other approach is to find frequent user-sets. The data can be pre-processed and converted into the following format,

SongID - User ID$_1, UserID_2, \ldots UserID_N$

for all the songs and then apply Apriori to it.

This allows us to find similar users. Songs listened by a user are then recommended to similar users.

Similar work [2] using Apriori algorithm has already been done using Apriori algorithm to solve the cold start problem.

4.2 Collaborative Filtering

The most basic models for recommendations systems are collaborative filtering models[5] which are based on assumption that people like things similar to other things they like, and things that are liked by other people with similar taste.

We plan to use the matrix factorization technique to generate user vector embeddings and song vector embeddings. For any given song, similar songs are then computed using cosine distance (We are not sure if our system will have enough computation power to calculate the cosine distance for all the songs and this could be a possible limitation). Similarly, for any given user, similar users can be figured out using cosine distance.

4.3 Clustering

Clusters of similar songs can be generated using various clustering algorithms such as k-means algorithm or DB Scan algorithm. The song vector embeddings obtained using Matrix Vectorization are used to generate these song clusters. For any given songs, songs from the same cluster are then recommended to a user.

4.4 Possible Extensions

The Million Song Dataset has content information about each track. We can get attributes like tempo, loudness, energy for the tracks. We can also obtain information about different artists associated with a track and generate similar artist clusters if required using song vector embeddings. These attributes when combined with other features can be used to form feature vectors which can then be used to form better song clusters and hopefully generate better song recommendations. This can even be used to tackle the cold start problem.

If we are able to finish the previous sub-tasks before time, we will experiment with these and compare the performance of it.

4.5 Feasibility

The work requires a lot of literature review from our side and understanding the best practices for our proposed work. Since we are graduate students, I believe we will be able to do this. The subtask of figuring out similar songs using matrix factorization might run into problems because of insufficient computation power of our systems. We might have to find an alternative in that case.

5 EVALUATION

Collaborative filtering is the state of the art technique for generating recommendations. We plan to implement it and the results generated from it would be our base results. The recommendations generated from our other proposed methods will be compared to these results and the correctness will be measured using that. We also aim to understand why one model works better than the other and figure out the limitations of it if possible.

If possible, we also plan to get our predictions evaluated by experts in the field of music research for verifying if our recommendations for new songs(from all different proposed tasks) are suitable or not.

6 TIMELINE

- Setting up environment: Week 8
  - Choose a machine for this project
  - Setup OS, packages, libraries
- Data Cleaning/Pre processing: Week 9 and 10
  - Data conversion for all versions of Apriori
  - Data conversion for collaborative filtering
- Data Mining Week 10 and 11
  - Aprypri, Python package, for apriori
- Project checkpoint: Week 12
  - Showing results of mined patterns and briefly comparing the models.
- Data Mining: Week 13 and 14
  - Surprise, Python package, for collaborative filtering
  - Clustering
- Evaluation and Project Report: Week 15 and 16

7 FINISHED TASKS

We are currently meeting all the tasks according to our timeline.

7.1 Data Preprocessing

- Information about the data: There are three major files that we are using, "*kaggle_songs.txt*" (has mappings of *song_ids* to numbers), "*kaggle_users.txt*" (has mappings of uids to numbers) and "*kaggle_visible_evaluation_triplets.txt*" (has triplets in each row which tells how many times a user listened to a particular song). We generated two types of datasets.
  The first type is:
  UserID - $songID_1, songID_2, \ldots songID_N$
  The second type is:
  SongID - $UserID_1, UserID_2, \ldots UserID_N$
  The UserIds and SongIds were later replaced by their corresponding numerical values. The transformed dataset has 163206 unique songs and 110000 unique users in it.
- Observations: We tried applying Apriori algorithm on the entire dataset. Our machines were not powerful enough to run the computationally expensive Apriori algorithm on all the data. We thus observed the distribution of the frequency of each song, so as to run Apriori on only some meaningful subset of the data and not all of it. The interesting thing that we observed was that around 154985 songs have their frequency in the range from 1 to 100. Out of these songs, 108054 have their frequency less than 10. This is a huge number. We can get rid of these songs because they are going to be infrequent and test our data on songs that have their frequencies in the range of 11 to 5000. There are 51808 such songs and we have now reduced the dataset by almost a factor of 3.
  We have attached all the interesting plots showing the distribution of song frequencies at the end of the report.
- Problems faced: We have huge amount of data with limited computation power. It is difficult to even generate transactions in the form of two itemsets mentioned above. We then approached the problem smartly to first sort the data and then generate itemsets.

7.2 Implementing Apriori algorithm

We tried implementing the apriori algorithm on the entire dataset using existing python libraries. We couldn't implement that because of limited computation power of our machine. However, we were able to apply it successfully on small subsets of the entire dataset and generated decent number of 2 frequent itemsets and 3 frequent itemsets.

7.3 Implementing Collaborative filtering

We are currently pre processing the data for collaborative filtering. We have already prepared the model for implementing it. Running it on the entire dataset will take a huge amount of time. We will use the reduced dataset to generate some base results to compare with.

8 FUTURE WORK

Here are the remaining and modified tasks that we will try to accomplish before the project deadline.

8.1 Exploring GCP and AWS

We are exploring these two cloud platforms to try and run our algorithm on the entire dataset and mine some novel patterns. We are new to these cloud platforms and this step will take some time.

8.2 Implement Apriori on reduced dataset

We plan to create the reduced dataset and then apply apriori on it and see its performance. We need to evaluate how useful are the mined frequent patterns.

8.3 Implement FP Growth or variants of Apriori on the entire dataset

Since there are modifications to the existing Apriori algorithm and entirely different algorithms like FP growth exist to speed up the process of finding frequent itemsets, we plan to try them out as well and see if we are able to mine any useful frequent patterns with limited computation power.

8.4 Implement Collaborative filtering

We plan to apply the matrix factorization method to get song vectors and compute the distance between them to figure out similar songs. We plan to compare the results obtained from CF method with those obtained from frequent itemsets.

8.5 Design the recommender system

We plan to combine all our observations to generate similar song recommendations for any song that is provided as an input.

8.6 Project Report

The last component will be to finish the project report and submit it along with all the code and dataset.

9 HONOR CODE PLEDGE

- On our honor, as University of Colorado Boulder students, we have neither given nor received unauthorized assistance.

10 SOME INTERESTING PLOTS

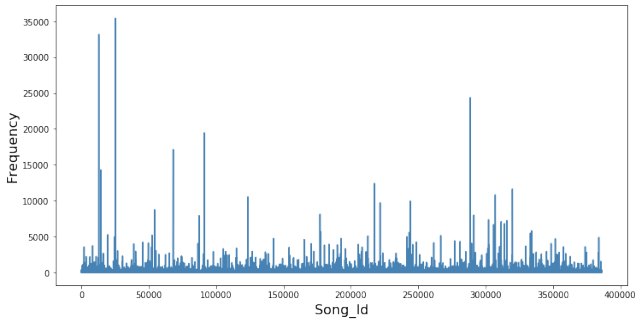

Figure 3: Plot of Song id with frequency

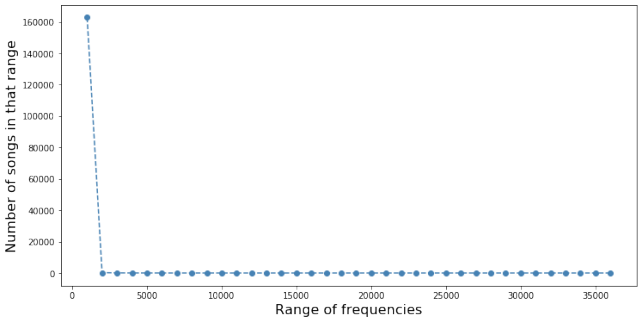

Figure 4: Plot of range of frequency vs number of songs in that range (bucket size=1000)

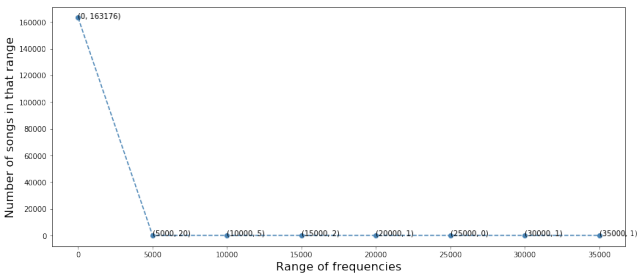

Figure 5: Plot of range of frequency vs number of songs in that range (bucket size=5000)

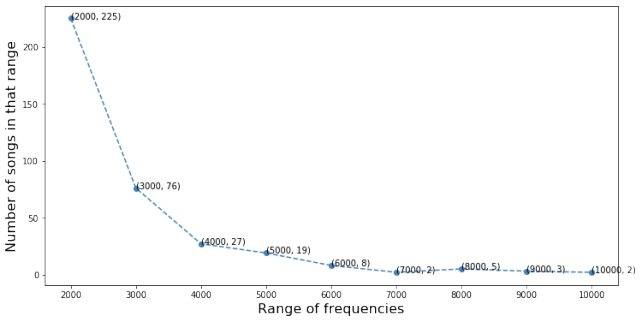

Figure 6: Plot of range of frequency vs number of songs in that range from 2nd to 10th bucket (bucket size=1000)

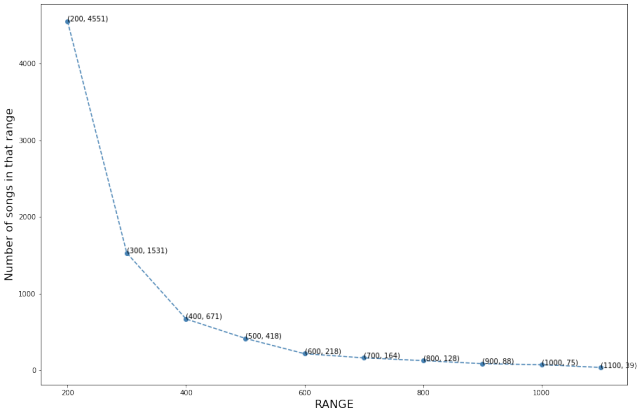

Figure 7: Plot of range of frequency vs number of songs in that range from 2nd to 10th bucket (bucket size=100)

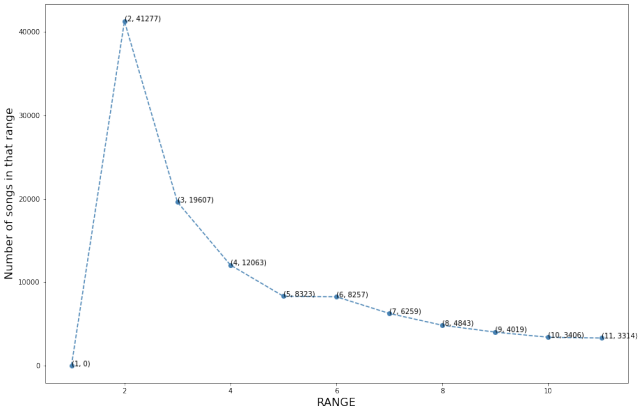

Figure 8: Plot of range of frequency vs number of songs in that range from 1st to 11th bucket (bucket size=1)

REFERENCES

[1] Natalie Aizenberg, Yehuda Koren, and Oren Somekh. 2012. Build Your Own Music Recommender by Modeling Internet Radio Streams. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/2187836.2187838

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *IJECS Volume 3 Issue 7 July, 2014 Page No.6855-6858.*

[3] Ò. Celma. 2008. *Music Recommendation and Discovery in the Long Tail.* Ph.D. Dissertation. Universitat Pompeu Fabra, Barcelona.

[4] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 714–722. https://doi.org/10.1145/2339530.2339643

[5] Prince Grover. 2017. *Details about Collaborative Filtering.* Retrieved October 10, 2019 from https://towardsdatascience.com/various-implementations-of-

collaborative-filtering-100385c6dfe0

[6] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware Music Recommendation Based on Latenttopic Sequential Patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems (RecSys '12)*. ACM, New York, NY, USA, 131–138. https://doi.org/10.1145/2365952.2365979

[7] Kaggle. 2012. *Kaggle Dataset*. Retrieved October 10, 2019 from https://www.kaggle.com/c/msdchallenge/data

[8] Noam Koenigstein, Gideon Dror, and Yehuda Koren. 2011. Yahoo! Music Recommendations: Modeling Music Ratings with Temporal Dynamics and Item Taxonomy. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 165–172. https://doi.org/10.1145/2043932.2043964

[9] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.

[10] Markus Schedl, Peter Knees, and Johannes Kepler. [n.d.]. Context-based Music Similarity Estimation. In *in Proceedings of the 3rd International Workshop on Learning the Semantics of Audio Signals (LSAS 2009*. 2009.

[11] Lohr Steve. 2009. *A 1Million Research Bargain for Netflix, and Maybe a Model for Others*. Retrieved September 21, 2009 from https://www.nytimes.com/2009/09/22/technology/internet/22netflix.html

[12] V.Manvitha1 and M.Sunitha Reddyand. 2014. Music Recommendation System Using Association Rule Mining and Clustering Technique To Address Cold start Problem. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2014)*.

[13] Wikipedia. 2019. *Details about Apriori Algorithm*. Retrieved October 10, 2019 from https://en.wikipedia.org/wiki/Apriori_algorithm