CSCI 5502
Music Recommendation System

Youngjin Ko (yoko2606@colorado.edu)
Yutaka Urakami (yutaka.urakami@colorado.edu)
Himanshu Gupta (himanshu.gupta@colorado.edu)
Jaeyoung Oh (jaeyoung.oh@colorado.edu)

Why recommend music?

- We all use various platforms for streaming music, and with lots of options to choose from, people sometimes feel overwhelmed.
- This raises the need for an efficient music recommender system for customers.
- With the help of such systems, companies can improve users' satisfaction and ensure high retention of users.
- A large scale, open and transparent Million Song Dataset gives us an opportunity to work on a music recommendation system that is centered around users and their behaviour (which is a hot topic these days).

Why this problem and challenges?

- 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.
- 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 is a challenging task. Comparing results with other new 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.

Data

We plan to use two existing datasets- Kaggle's dataset and Million Song Dataset.

Kaggle dataset includes:

- 386,213 song IDs and 110,000 user IDs
- each user's listening history (user_id, song_id, #listening) in kaggle_visible_evaluation_triplets.txt
- relationship b/w song_id and track_id(s) in taste_profile_song_to_tracks.txt

Million Song Dataset includes:

- each track information (tempo, loudness, energy and so on) in track_id.h5
- List of all artist ID (artist_id, artist, mbid, track_id, artist name) in unique_artists.txt
- And so on!

Previous Work

- Build Your Own Music Recommender by Modeling Internet Radio Streams, WWW 12, [1]
  - Goal: Personalized music recommendation
  - Data: playlists of thousands of Internet radio stations (Millions of plays, hundreds of thousands of tracks and artists)
  - Method: a probabilistic collaborative filtering (CF) model

- Yahoo! Music Recommendations: Modeling Music Ratings with Temporal Dynamics and Item Taxonomy, RecSys 11, [2]
  - Goal: Improve Yahoo music recommendation accuracy
  - Data: Yahoo music dataset, million users, 600 thousand musical items, 250 million ratings for decade
  - Method: matrix factorization with temporal analysis of user ratings, and item popularity trends

Previous Work

- Playlist Prediction via Metric Embedding, ACM KDD 12, [3]
  - Goal: automated playlist generation
  - Data: radio playlist from yes.com (hundreds of stations in the U.S)
  - Method: Latent Markov Embedding (LME)

- Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns, Recsys 12, [4]
  - Goal: Context-aware music recommender system
  - Data: human-compiled music playlists (7,051 playlists, 21,783 songs)
  - Method: Topic modeling with the latent topics generated from the the most frequent tagged songs.

Proposed work and techniques

- We have already seen how identifying frequent itemsets helps is generating item recommendations for users. Why not use the same algorithm for generating song recommendations?

- If possible, use audio content features to cluster similar songs and then generate relevant recommendations from that.

- Tackle the cold start problem.

Evaluation

- We plan to compare and evaluate our model against the state of the art collaborative filtering technique and compare the recommendations.
    - If possible, even understand why one model works better than other.

- If possible, we plan to get our predictions evaluated by experts in the field of music research for verifying if our recommendations for new songs are suitable or not.

Finished Work (till now)

- Data Pre-processing
    - Generated two types of itemsets.
    - Type 1 <uid> - {song_id1, song_id2.... Song_idn}
    - Type 2 <song_id> - {uid1,uid2.....uidn}

    - Problems faced -
        - Huge amount of data with limited computation power.
        - Devised an approach to first sort the data and then generate itemsets.
        - Repeated this for generating both types of dataset

    - NUM_UNIQUE_SONGS = 163206
    - NUM_UNIQUE_USERS = 110000

Finished Work (till now)

- Explored alternate options to get session ids and generate sequences but no such data was found.

- Implementing Apriori Algorithm to determine frequent patterns
    - Tried implementing it using existing Python libraries
        - apyori (https://pypi.org/project/apyori/)
        - mlxtend.frequent_patterns
    - Problems faced
        - Huge dataset and limited computation power (once again).
        - These libraries seem to have some issue when the item names are random strings(as is in the dataset).
    - Proposed Solutions
        - Use a numerical mapping for encoded Uid and Song_id.
        - Use FP-Growth instead or get a GPU.

Finished Work (till now)

- Implementing collaborative filtering.
  - Finished data pre processing for the dataset to get data in the proper format for applying CF.
  - Implementing a basic CF model using existing tools.

Future Work

- Determine Frequent Itemsets using Apriori or FP Growth on the dataset.
- Mine strong association rules and generate recommendations for any given song.
- Apply Weighted Apriori and compare the results with normal Apriori.
- Generate recommendations from collaborative filtering and compare our model's recommendations.
- Compare if Weighted Apriori had any benefits over Normal Apriori while generating recommendations.

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.
2) 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
3) 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.
4) Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware music recommendation based on latent topic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems* (RecSys '12). ACM, New York, NY, USA, 131-138