CS 6220: Project Proposal

Jason Miller, Shefali Khatri

Title of Project:

Song Recommendation System

Purpose:

- Recommendation systems enhance the user experience for both the seller and the consumer. This is for many contexts. The seller could be a brand, a content producer, an advertisement agency, etc. The consumer could be a buyer of some products, consumer of some type of media, “viewer” of advertisements. The recommendation system enhances the user experience for the seller because their product is able to better reach the targeted seller, while it enhances the user experience for the consumer because the consumer does not need to spend time sifting through irrelevant/undesirable products.
- Further, from the perspective of a consumer, recommendation systems are valuable because they encourage consumers to understand and diversify their interests in certain products. Consider Spotify’s song recommendation system. The recommendation system provides users with a more efficient pipeline to discover what kinds of music they may like. This is especially valuable when the product itself is some sort of creative art.

Proposed Approaches:

Based on our research, we believe that one of the following approaches would enable us to build a highly predictive recommendation system:

1. Content based filtering - This approach uses user preferences and their previous history to determine new songs to recommend.
2. Collaborative filtering - This approach measures similarity amongst users to determine new songs to recommend. We could either use deep learning or a matrix factorization based algorithm.
3. Multi-Armed bandit methods - This approach can be used in combination with other approaches to address the cold-start problem. The cold-start problem arises when new users join a platform. As new users have yet to generate any data, a multi-armed bandit will enable us to increase the likelihood of a new user being recommended songs they would enjoy.
4. Neural Networks - In 2016, Google published a paper highlighting the recommendation system used for Youtube. This recommendation system consisted of 2 neural networks. One was used for generating a set of candidates, while the second neural network was responsible for ranking candidates to recommend. We hope to utilize autoencoders and embeddings in our networks.

Additionally, we can employ some combination of the above approaches or try to explore using ensemble methods to generate a recommendation system. Our analysis suggests that most companies using recommendation systems today do not have one approach but a combination of the above approaches to determine content to recommend. Most recommendation systems follow a specific framework composed of 3 parts:

1. Candidate Generation - Reduces recommendations to a small subset of candidates. More than one candidate generation system may be used.
2. Scoring Systems - Candidates are scored and ranked based on another model
3. Re-Ranking (Optional) - The candidates can be re-ranked using additional constraints such as filtering which may reduce the candidate set.

Project Plan:

Week 1 Tasks:

Exploratory data analysis, feature engineering, handle outliers, missing data, and explore a dimensionality reduction technique to use such as PCA, SVD, NMF, autoencoders.

Week 2 Tasks:

Finalize on dimensionality reduction technique, start exploration on an approach for candidate generation. Develop and explore the use of embeddings in our neural network.

Week 3 Tasks:

Finalize candidate generation and begin exploration on ranking system. Evaluate various approaches.

Week 4 Tasks:

Evaluate various approaches and select the final recommendation system. Formulate a paper discussing our findings and begin preparation for our presentation.

Data Source:

The data used for this project will be sourced from the following sites:

- Million Song Dataset: Welcome! - a collection of metadata for a million popular music tracks.
- The Echo Nest Taste Profile Subset - user data that contains track play counts
- thisismyjam-to-MSD mapping - more user data that contains track play counts

The datasets containing user data will be paired to the MSD using the song ID's. The data for this project has previously been used as part of the Million Song Dataset Challenge on Kaggle as well as numerous studies on recommendation systems.

Proposed Evaluation:

- HR (Hit Ratio) - This is simply the fraction of users for which the correct answer (recommendation) is included in a recommendation list of length *L*. It's important to choose an appropriate value of *L*.

$$HR = \frac{|U_{hit}^{L}|}{|U_{all}|}$$

where $|U_{hit}^{L}|$ is the number of users for which the correct answer is included in the top L recommendation list, $|U_{all}|$ is the total number of users in the test dataset.

- MRR (Mean Reciprocal Rank) - This is also known as the average reciprocal hit ratio. There are different variations or simplifications for calculating the relevance rank, *RR(u)*. For an implicit dataset, which we need to decide whether or not we have, the relevance score is either 0 or 1, for items not bought or bought (not clicked or clicked, songs not listened to or listened to).

$$MRR = \frac{1}{|U_{all}|} \sum_{u=1}^{|U_{all}|} RR(u)$$
$$RR(u) = \sum_{i \leq L} \frac{relevance_i}{rank_i}$$

where $RR(u)$ is the reciprocal rank of a user u, and it is defined by the sum of relevance score of top L items weighted by reciprocal rank. MRR is simply the mean of all users in the test dataset.

- MAP (Mean Average Precision) - We can calculate the average precision for each user and take the mean over all users to calculate MAP. We need to compute precision and recall at every position in a ranked sequence of documents to plot a precision-recall curve and then we can calculate the area under the precision-recall curve to determine average precision.

$$AP = \sum_{k=1}^{n} p(k) rel(k)$$

where $p(k)$ denotes precision@k, n is the number of items in the recommendation list, and $rel(k)$ is an indicator function which equals 1 if the rank k item is relevant, 0 otherwise.

$$MAP = \frac{1}{|U_{all}|} \sum_{u=1}^{|U_{all}|} AP(u)$$

- NDCG (Normalized Discounted Cumulative Gain) - Gain for an item is essentially the same as a relevance score, and cumulative gain (CG) is defined as the sum of gains up to a position *k* in a recommendation list. Cumulative gain does not account for ordering, and we can compensate for this using discounted cumulative gain (DCG) which penalizes highly relevant items being placed at the bottom. DCG is not a consistent metric to compare recommendation lists of differing lengths because lists of longer lengths will have a higher score simply because of its length. We can account for this by using ideal DCG (IDCG). We then generate the normalized discounted cumulative gain so that we have values between 0 and 1 regardless of length of the recommendation lists.

$$DCG(k) = \sum_{i=1}^{k} \frac{G_i}{log_2(i+1)} \qquad IDCG(k) = \sum_{i=1}^{|I(k)|} \frac{G_i}{log_2(i+1)}$$

$$NDCG(k) = \frac{DCG(k)}{IDCG(k)}$$

Resources used: Multi-armed bandits for dynamic movie recommendations | by Brian O'Gorman | Insight (insightdatascience.com); How YouTube Recommends Videos. This is the first paper in a “Seminal… | by Moin Nadeem | Towards Data Science; Retrieval | Recommendation Systems | Google Developers; Ranking Evaluation Metrics for Recommender Systems | by Benjamin Wang; Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering | by Abhijit Roy | Towards Data Science