Vahe Avagyan - 49128016, Prateek Verma - 84162060, Kartik Saxena - 84568218.

Project ID: pp15

**CS277:Music Recommendation Engine Project Report**

**Abstract.**

We implemented a music recommender system based on user’s listening history and item based similarity. We used the Million Song Dataset (MSD) to find correlations between users and between songs to ultimately provide recommendations for songs to which users would prefer to listen. In particular we used the EchoNest user profile data. We discuss the challenges we faced, the methods we implemented for this system, and the results and analysis of our observations. We compute user based similarity using actual play count in the listening history and the item based similarity using common users, which listened to those songs. Our final recommendation is based on the combination of these two results. Our method is to give user-based song recommendations to certain users and item based recommendation to others based on certain properties of that user’s listening history. We specifically use the length of the user’s playing history above or below a certain threshold as a decision parameter.

1. **Introduction.**

With the increase of commercial applications and cloud services offering music, there is a need to present the user, relevant and personalized music recommendations, taking into account user centric data, preferences and semantic annotations. Most of the academic work is based on only automatic semantic annotations ignoring user centered data, primarily due to lack of openly available datasets and intellectual privacy concerns. The Million Song Dataset [1], however, addresses this concern providing the research community an open and large scale data with listening histories of over a million of users, meta-data for music, audio content-analysis, and standardized identifiers.

There are two ways to build a recommendation engine, using collaborative filtering or using content based filtering based on acoustic data. Collaborative filtering is a method of making automatic predictions based on the preferences of many users. It basically explores technique of matching people based on similar preferences and making recommendation on that basis. Most commercial engine like iTunes, Amazon, etc. use collaborative filtering, where annotations are done through crowdsourcing using online interactions, human experts, and games that could be designed to collect tags. The biggest challenge in a such recommendation systems is identifying similarity between data samples and evaluating the resulting predictions [1][3][4]. Another key issue in designing music recommendation system is that two similar songs can have very different audience, considering this fact, selecting the features of the songs is important.

Our source of data is Million Song Dataset and EchoNest dataset of user Taste Profile [2]. Thus, there are two kinds of data, user listening data and song metadata. The MSD provides data about most popular music tracks and contains meta-data that allow identifying each track. The meta-data contains song information such as song name, song ID, artist name,- year composed, and some audio signal related measurements. The EchoNest dataset includes 48 million triplets [user, song, count] collected from listening histories of 1.2 million users. The user playcount data represents the number of times a user has listened to a particular song. The EchoNest dataset has 380,000 songs from MSD. We have used this data as a weight value for user based collaborative filtering using different approaches, such as k-Nearest neighbors, hierarchical clustering, and cosine similarity models to get information about user preferences. We form a matrix between users and songIDs with corresponding elements filled as play counts from EchoNest dataset. This results in a very sparse matrix, that we use for the methods we have implemented.

1. **Problem Definition.**

The goal is to generate a song recommendation for a user given a part of his play history with play counts. The recommendation should be generated using dataset that includes a set of users with their full history. The challenge here is to get information from given histories for measuring the similarity between songs and their relevance to the given user. A common approach for such problems is collaborative filtering. The idea of collaborative filtering is to treat the song play history of a user as a vector that describes the user

Another challenge is to process big amount of data from EchoNest dataset to get the full user and song features and increase the prediction accuracy. Since it was impossible to process all the data provided from EchoNest dataset, we took a randomized set of thousand entries in EchoNest dataset for our experiments.

1. **Method.**

Before implementing our algorithms for the recommendation system we have loaded the MSD dataset to our working environment and represented it in a more manageable way. Thus, having triplets of *<userID-SongID-playCount>* we created an array where rows represent unique users and columns are unique songs. The values in the matrix are *playCounts* for each user-song pair. Next, we normalize *playCount* values to represent the data in the range of [0,1] and store the resulting matrix as a sparse matrix to use memory more efficiently.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | userID | SongID | Count | | User1 | Song 1 | 1 | | User1 | Song2 | 1 | | User2 | Song3 | 5 | | User3 | Song2 | 2 | | User3 | Song4 | 8 | |  | … |  | | Training Data  70%  Test Data  Evaluation Data  2042 users  44458 songs  Figure 1: Data representation. |

We divided user listening history data into training and testing data. For training data we took 70% of the users with their full listening history, while for testing data we used the remaining 30% and remove the second half of user history. The first part of history in testing data is used for initial identification of user preferences that is needed to predict the second part using the model.

For the music recommendation system we developed several algorithms based on collaborative filtering, which we will describe below. While there are many different methods of implementing collaborative filtering, we used k-Nearest Neighbor algorithm, hierarchical clustering, user based, item based collaborative filtering, and a mixed method that combines user based and item based methods. For each algorithm song play counts were used as weights to define how relevant the given song to the user. List of the most relevant songs for the given user is sorted by play counts and outputs as final recommendation.

**3.1. User-Based collaborative filtering.**

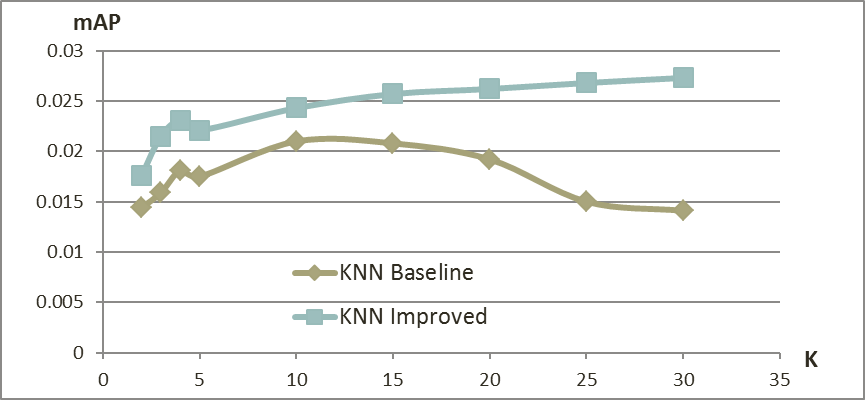
User based collaborative filtering is an approach to recommend items based on items (in our case) songs listened by the users similar to the test user. The idea is that, the test user is more likely to listen to songs if some user who is similar to it has already listened. We use various algorithms to determine similarity between users.

**K-Nearest Neighbor.**

We implemented a recommendation system using k-Nearest Neighbor (kNN) algorithm as a baseline method. We used kNN algorithm to train the model and predict songs for the missing part in the testing data. For a given user the algorithm finds k users with the most similar preferences. The similarity between users is based on the number of common songs in their listening histories and computed by kNN algorithm using cosine distance between them. In order to recommend songs for a given user from the test data the algorithm picks a matching user with the longest play list. The intuition of this approach is that the user with longer history will provide more songs for recommendation.

The next step was improving this method by combining playlists of k most similar users, sorting them by play counts, and selecting top 10 songs that do not occur in user’s history for the recommendation. The result of this result is correlated with the number k, while the result of baseline method is unpredictable and does not have any relation with k.

Figure 2: Baseline and improved kNN approach comparison.



**Hierarchical clustering.**

The idea of hierarchical clustering is to combine similar users together into one cluster and maintain a hierarchy of such clusters. At the first step the algorithm treats each user as an individual cluster. Next, it computes the cosine similarity between them and chooses pairs with the highest score to create a cluster for each initial cluster. A user can be clustered with another user or with some cluster produced on previous steps. This allows creation of hierarchy of clusters, where the first level of clusters is combinations of the most similar users. To find more than one similar user to the given user we implemented an algorithm that traverses up to the hierarchy and returns k similar users. This allows increasing the number of songs for recommendation.

**Cosine similarity measure.**

Another and the most common approach for user-based collaborative filtering is computing similarity between users. In this method features for each vector is its listening history.

**3.2. Item-Based collaborative filtering.**

Item based collaborative filtering is the approach of recommending songs based on songs similarity. The algorithm takes the songs that the test user has already listened to and computes item based similarity using cosine distance. The feature vector for a song is the set of users that have listened to the song. After aggregating all the songs similar to the songs already listened we recommend only songs having top play count sums.

**3.3. Mixed model**

The intuition behind the mixed model is to use different algorithms depending on the length of user’s listening history. Thus, we defined a threshold that is used to decide which method to use for a given user. We generated a validation data in order to test our algorithm performance and tune the threshold. The algorithm computes similarities between pairs of songs by using cosine similarity. This approach considers the number of users that have both songs from the pair in their histories and does not require any additional metadata about songs. For the case when the length of the listening history is above the threshold we used user based approach, which computes cosine similarities among all users and recommends songs from aggregated playlist. The similarity measure is based on the number of common song in users’ playlists. For a given user the algorithm aggregates playlists of k most similar users and selects the most similar songs to the songs that the user has in its list. For song selection from the playlist the algorithm uses computed cosine similarities between songs that are used in item based collaborative filtering.

Recommendation Model

User-Based Similarity Metric

Item-Based Similarity Metric

If playlist length is less than threshold

If playlist length is higher than threshold

Figure 3: Mixed model structure.

1. **Results / Experimental Evaluation**

**4.1 Methodology**

We used the truncated mAP( mean average precision) as an evaluation metric. Let y denote a ranking over items, where y(p) = i means that item i is ranked at position p. The mAP metric emphasizes the top recommendations. For any k , the precision at k (k) is defined as the proportion of correct recommendations within the top-k of the predicted ranking (assuming the ranking y does not contain the visible songs),

For each user the (truncated) average precision is the average precision at each recall point: where u is the smaller between  and the number of user u’s positively associated songs. The average of AP(u; yu)’s overall users gives the mean average precision (mAP).

**4.2 Results**

The performance of our algorithms is presented in Table 1. The table shows that the best performance is obtained by user based collaborative filtering using cosine similarity.

Table 1: mAP score results.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **KNN Baseline** | **KNN**  **Improved** | **Hierarchical Clustering** | **User based cosine** | **Item based cosine** | **Mixed model** |
| mAP | 0.0141 | 0.0273 | 0.0167 | 0.0348 | 0.0015 | 0.0166 |

Item based collaborative filtering using cosine similarity gives the lowest score comparing to other methods. This can be explained by incompleteness of the song feature vectors. While for user based collaborative filtering we have the full user listening data, item based method is using only the data from the subset we retrieved from EchoNest dataset.

The baseline KNN approach has low mAP score since it is choosing only one user with the longest play history, which may not contain the most relevant songs. Therefore, the result is unpredictable and not always relevant. Improved kNN algorithm that aggregates playlist of k similar users gives much better prediction and it is score is dependent on the value k. The comparison between two kNN approaches is shown on the graph.

Hierarchical clustering does not perform well for this problem, since the algorithm takes k clusters to which the given user belongs for generating the prediction. While the most similar users are in the first cluster, taking clusters from the higher level gives more general data. Thus, the obtained song data from the higher cluster is relevant to all the users evenly, but does satisfy specific preferences of the given user.

1. **Related Work**

A lot of related work has been completed on the Kaggle competition site relating to the Million Song

Dataset back in 2011 when the challenge was released for the public. Although the specific implementations of the competitions various solutions were not revealed, various of algorithms were used by different teams for this purpose. There have been a lot of published works as well, regarding the different methods to evaluate the data to produce song recommendations. Some of this work  and its different aspects are discussed here in this section. There are a number of different ways to measure the similarities between songs, such as semantic embedding model [5] and acoustic similarity model [6]. However, music similarity measures are subjective, which makes it difficult to rely on ground truth. This issue is addressed in [7] and [8].

Meta-data provides rich information for the songs, which helps classify the music. However, meta-data suffers from a popularity bias. People are more likely to rate the artists and songs that are popular. Thus, little information can be found for those less known artists and songs. [9].

Numerous other methods have been considered and implemented in [10]. One method that is very similar to SVD approach is called Latent factor mode Bayesian personalized ranking (BPR).

1. **Scalability.**

KNN , Collaborative Filtering Algorithms can be implemented on top of Apache Mahout library which is based on the MapReduce paradigm.

Map-Reduce can be also used for pre-processing and dataset creation.

The algorithms we used have algorithmic complexities shown in Table 2.

Table 2: Complexity of algorithms.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **KNN Baseline/ Improved** | **Hierarchical Clustering** | **User based cosine** | **Item based cosine** | **Mixed model** |
| Complexity | O(N2) | O(N3) | O(N2) | O(M2) | O(M2) |

Cosine similarity is a common technique in machine learning that can be computed using map reduce. With mapReduce approach, incoming data is partitioned among processing nodes.

In each processing node, songsIDs for given user are sequentially processed and the data is collected in maps. Next, this data is redistributed using the portioning scheme for reducers, thus mappers emit songs as keys for reducers. At the next step reducers merge the data incoming from the mappers. Then, at the local level contribution of each song to the similarity value is calculated for the users where it occurs. All the results are then sent to one node and partial results are merged.

1. **Conclusion.**

In this study the user-based and item-based collaborative filtering algorithms had a better recommendation accuracy as compared to the baseline approach using solely popularity of songs. We feel that both of these approaches can be applied in a music related recommendation service to provide music listeners with a significantly improved recommendation accuracy. Both of these algorithms can be applied in music services with pre-selected tracks for users which can significantly enhance the usability of such systems. Another future task for us would be to evaluate the recommendation performance taking play-counts into account , currently we only use the binary information whether the track appeared in the user’s listening history or not. Another approach to be explored is to calculate item-based similarity using song based similarity and calculating item similarities using song based metadata in the Million Song Dataset[1]. The data set is a freely available collection of audio features and metadata for a million contemporary popular music tracks.

Our work looked at leveraging mostly user listening behavior and song characteristics to identify similarities between the tracks. Generally, further work can be done to develop more recommendation algorithms based on different data (e.g. the how the user is feeling, which www.ragechill.com tries to capture), and these new methods can be used to further enhance recommendation precision.

It would be also interesting to investigate how recommendation precision change leveraging social network relationships into account. The intuition here is that users generally have similar listening habits as their friends and perhaps recommendations can be influenced by recommending songs from their friend’s listening history.

The precision attained in the item-based similarity method was quite low to our surprise. After carefully analysing the dataset again we realized the reason for it. Due to the fact that we only considered the first 100k tuples out of the 48 million tuples , it did not represent true representation of songs being played by users. When computing item based similarity scores , the scores were much lower as the the number of users who listened to these tracks were low. This resulted in low cosine similarity values between items and led to poor recommendations and lowered the mAP value. We believe that a very effective method of music discovery lies in compiling the results from a variety of different individual recommendation algorithms, like the results from our mixed - model have indicated. The results were lowered due to the poor scores we got from item based similarity , but overall the results of the mixed model are satisfying.

These shortcomings perhaps would be overcomed by simple enhancements such as taking a much larger data-set for evaluation. For example in the first 100k tuples we had around 44458 unique songs and 2042 users while the original dataset had around 300,000 songs and 1 million unique users. Clearly in our dataset we have less users overall as compared to the number of songs which leads to poor calculations for item based similarity as it depends on the number of common users who listened the two tracks between which the similarity is being computed.

**References.**

[1] Million Song Dataset, official website by Thierry Bertin-Mahieux, available at: http://labrosa.ee.columbia.edu/millionsong/.

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference(ISMIR 2011), 2011.

[3] http://sifter.org/\_simon/journal/20061211.html

[4] Sergey Brin, Lawrence Page.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, Volume 30, Issues 17, April 1998, Pages 107117

[5] Law, E., Settles, B., Mitchell, T.: Learning to tag from open vocabulary labels. In: ECML. (2010)

[6] Bertin-Mahieux, T ., Eck, D., Mandel, M.: Automatic tagging of audio: The state-of-the-art. In Wang, W., ed.: Machine Audition: Principles,Algorithms and Systems. IGI Publishing (2010) In press.

[7] Berenzweig, A.: A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures. Computer Music Journal 28(2) (June 2004)

[8] Ellis, D.P.W., Whitman, B., Berenzweig, A., Lawrence, S.: The quest for ground truth in musical artist similarity. In: ISMIR. (2002)

[9] Pampalk, E., Dixon, S., Widmer, G.: On the evaluation of perceptual similarity measures for music. In: Intl. Conf. on Digital Audio Eects.(2003)

[10] Law, E., West, K., Mandel, M., Bay, M., Downie, J.S.: Evaluation of algorithms using games: the case of music tagging. In: Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR). (October 2009)

[11] Fabio Aiolli.: Preliminary Study on a recommender system for the Million Songs Dataset challenge. (2011)

[12] Kaggle Million Song Dataset Challenge description,<http://www.kaggle.com/c/msdchallenge>

[13] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, The yahoo! music dataset and kdd-cup '11, KDD-Cup Workshop, 2011.

[14] Oscar Celma and Paul Lamere, Music Recommendation and discovery revisited, 2011. ACM Conference on Recommender Systems.