Music Recommendation System

Amin Amiripour, John Niolo

Department of Computer Science
University of Southern California, Los Angeles, USA
amiripou@usc.edu, jniolo@usc.edu

***Abstract*— In this project we have designed, developed and analyzed a "Music Recommendation System". The idea is to recommend the most popular songs to a user learning from his/her listening history. The aim is to suggest a set of songs to a user given half of his/her listening history and complete listening history of other 1 million people. We have designed different algorithms, employed them in our recommendation system, discussed their advantages and limitations and conducted experiments to analyse the efficacy of each of our algorithm. From our experiments we concluded that collaborative filtering techniques are most suited for music recommendation systems.**

I. Introduction

Today we have different online music streaming services like Pandora[2], Spotify[3], Beats Music[4], last.fm[5] etc each of which generate revenues by offering music streaming services to its users. All of these music streaming service providers additionally recommend new songs to a user based on his/her music interests and listening history to further increase their revenues. Though all of these music streaming services recommend new songs to their users, but the approaches used to develop their music recommendation systems are different. For example, while last.fm uses user-based collaborative filtering techniques, Pandora makes use of item-based collaborative filtering techniques. Therefore, it can be said that there is no clear cut solution to develop a music recommendation system and different approaches can be used to develop one.

One important thing to note is that developing a music recommendation system is far more challenging than developing a recommendation system of an e-commerce website [10] [18]. In ecommerce space, if two items are being frequently bought together, then they can be recommended to a user if he/she already bought one of them before. But, two songs being frequently listened together don't imply that a user would like the other song if he/she has already listened to one of the songs. A lot of factors like the artist of the song, lyrics and acoustics of the song, mood of the user etc determine the liking of a song by a user. Hence, it's not straightforward to recommend songs to the users and care should be taken to include personal context of each user before recommending songs to him/her.

In this project we have implemented Instance-based learning algorithm (K-Nearest Neighbour) and Collaborative filtering approaches (both user-based and item-based) which are used to develop our music recommendation system. In the next section we describe the dataset we have used to build the recommendation system. In section III we talk about the algorithms we have implemented and in section IV we describe the experimental setup and analyse the results of our experiments. Finally, we conclude by suggesting some future work that can be done to further improve the accuracy of our recommendation system.

II. Dataset Description

We have used Million Song Dataset (MSD) [9] provided by Kaggle [1], a platform for data prediction and analytics competitions in our experiments. The data set consisted of listening history of a million users with three fields namely <user_id, song_id,play_count> format.

The dataset given by Kaggle for the MSD Challenge consisted of the listening history of million users in the form of triplets. Due to our experiment setup memory constraints, we used only a part of the data for our project. We have divided Kaggle dataset into 'training data' which is used to build the recommendation system and 'test data' which is the input to the recommendation system. The 'test data' is further divided into 'test visible' and 'test invisible' data. The 'test visible data' is used as the input data consisting of partial listening history of a user and with the help of recommendation system; songs that can be suggested for the user are predicted. The predicted songs are then compared with 'test invisible' data to check for the accuracy of the prediction of our system.

III. Algorithms

In this section we discuss the two algorithms we have used to develop our recommendation system. The two algorithms used by us are K – Nearest Neighbours algorithm [7] [12] and Collaborative Filtering Approaches (both user-based and item-based) [6] [11].

A) K- Nearest Neighbours Algorithm

K-NN is a popular technique often used to build recommendation systems. K-NN is useful in cases where one has little or no prior knowledge of the distribution of data. Kamal Ali et al. developed "TiVo" [20] which is a tv show recommendation based on K-NN that had 95% confidence interval value of 0.75-0.85. Owing to the simplicity of K-NN we have also used it to develop our recommendation system. In this algorithm for each of the test user we find 'K' users that have similar music interests as that of the test user, merge their listening histories, order the songs in the decreasing order of play count and return the top 'N' results. The intuition behind this approach is that similar users will have

similar interests which can be used to recommend music to users. The steps of our algorithm are described below.

- Initially we find a feature vector for all of the users present in the dataset. The feature vector for each user consists of their corresponding play counts of all of the songs present in the dataset and listed in order.
- Closeness between test user and other users is determined by finding the cosine similarities of their feature vectors.
- 'K' users that give highest cosine similarities are listed and all of their listening histories are merged to form the recommendation list for the 'test user'.
- Finally, the recommendation list is sorted in decreasing order of the play counts and 'N' top songs are returned as the recommendation songs for the 'test user'.

B) Collaborative Filtering Techniques

Collaborative Filtering (CF) techniques use the items by user matrix to discover other users with similar tastes as the active user for whom we want to make the prediction. The fundamental assumption of CF is that if users 'X' and 'Y' rate 'n' items similarly, or have similar behaviours (e.g., buying, watching, listening), then they will rate or act on other items similarly. The intuition is that if other users having similar interests as that of the active user already purchased a certain item, then it is highly likely that the active user will like that item as well. Collaborative Filtering techniques can provide unexpected findings by analysing the information about relevant users which is not true with respect to content-based filtering techniques [14] [15] and is one of the most successful recommender technology to date [16] [17]. This is the reason for choosing CF technique for building our recommendation system. There are different types of CF techniques that can be used to build a recommendation system. They are memory based CF, model based CF and hybrid model approaches. In our project we have used memory-based collaborative filtering technique since it is able to deal with large datasets in an efficient and effective way. Memory-based collaborative filtering is a technique to predict the item based on the entire collections of previous ratings. Every user is grouped with people with similar interests, so that a new item is produced by finding the nearest neighbour using a massive number of explicit user votes. There are two variants of Collaborative Filtering. They are user-based CF and item-based CF. We have implemented both variants of CF in our project.

1) User -based CF

The idea behind this algorithm is that similar users will have similar interests and hence similar listening history. So given a test user, by finding other users in the recommendation system that are similar to the test user, the listening histories of the matching users can be used to recommend songs to the test user. The basic algorithm is described in three steps as follows.

- Firstly, we build the user-to-user similarity matrix using cosine similarity for all of the users present in the dataset.
- Now, for any test user 'X' find a set of users from the user-to-user similarity matrix that closely match the test user with respect to the listening history. Then merge the listening histories of all of the matched users excluding the songs from the listening history of the test user to form a list of recommendation songs for the test user.
- Finally rank all of the songs in the recommendation list based on their similarity scores and return the top 'N' songs as the recommended songs for the test user.

The similarity between two users say 'X' and 'Y' is calculated by finding the cosine similarity of two users. It is intuitive that a user listening to lots of songs should be given a lower weightage compared to a user who has a small listening history when finding out the cosine similarities. This is achieved by using a weight factor 'α'€ [0, 1) [8].

$$Similarity(X, Y) = \frac{\#\ of\ common\ items\ between\ X, Y}{(\#\ of\ items\ of\ X)^{\alpha}.\ (\#\ of\ items\ of\ Y)^{(1-\alpha)}}$$

To calculate the similarity score for each song 'S' that needs to be recommended to user 'X', we find the sum of user-similarity scores of all users 'Y' (who belong to the train dataset)who have listened to the song 'S'. We have also used a normalization factor 'ϒ' [8] to maximize the impact of higher weights and minimize that of lower weights.

$$Weight_{(S)} = \sum_{(Y\ \epsilon\ train\ dataset)} Similarity(X, Y)^{\Upsilon}$$

After finding the score of all of the songs that could be recommended to user 'X', we finally recommend top 'N' songs with highest scores.

2) Item-based CF

The idea behind this algorithm is that a user tends to listen to songs that are similar to one another with the similarity being with respect to genre, artist or acoustics of the song. So when a user 'X' listens to song 'S' then the likelihood of him liking another song 'T' similar to 'S' is high and hence we can recommend the song 'T' to user 'X'. In fact Sarwar Badrul et al. have shown [19] that item-based algorithms dramatically provide better performance than user-based algorithms. The item-based algorithm implemented by us is described below.

- Firstly, we build the song-to-song similarity matrix using cosine similarity for all of the users present in the dataset.
- Now for user 'X' for every song 'S' that is present in his/her listening history, find other similar songs from the recommendation system by making use of song-to-song similarity matrix. Finally, find the similarity scores for all of the songs that could be recommended to user 'X' and return the top 'N' songs.

To find the cosine similarity between two songs 'S' and 'T', we again use the same formula we used in user-based CF for finding similarities.

$$Similarity(S, T) = \frac{\#\ of\ common\ users\ for\ S, T}{(\#\ of\ users\ for\ S)^{0.5}.\ (\#\ of\ users\ for\ T)^{0.5}}$$

To calculate the similarity score for each song 'S' that needs to be recommended to user 'X', we find the sum of song-similarity scores of all songs present in listening history of 'X' with that of song 'S' and return the top 'N' songs that have the highest similarity score as the recommended songs for user 'X'.

$\text{Weight}_{(S)} = \sum_{(T \in \text{listening history of test user})} Similarity(S, T)$

IV. Experimental Setup, Algorithms Evaluation and Results

In this section we briefly describe the experimental setup used to test our algorithms followed by the methodology we have used to evaluate the performance of our algorithms and conclude with the analysis of the results and findings of the experiment.

A) Experimental Setup

All the experiments were run on a single machine, whose configuration was 8GB RAM, Intel(R) Core 'i5' @2.50 GHZ.

B) General Methodology for Evaluation of Algorithms

The different steps involved in the evaluation of each of the algorithm are as follows.

- Generate three sets of data namely 'Test Visible Data', 'Test Invisible Data' and 'Train Data' from the original dataset.
  - Train data: Complete listening history of users used to train the recommendation system.
  - Test Visible Data: Part of the listening history of user using which the remaining listening history of the user needs to be recommended by our recommendation system.
  - Test Invisible Data: Remaining listening history of the test user which needs to be predicted by the recommendation system.
- Use the 'Train Data' to build the recommendation system.
- Use the 'Test Visible Data' and recommendation system to recommend the songs to the users.
- Evaluate the algorithm performance using 'Test Invisible Data' for a user along with the recommended songs for a user and find the precision for this user.
- Average Precision value obtained for all of the test users represents the algorithm's performance.

C) Precision Metric

We find the precision for each run by finding the ratio of number of recommended songs matching the 'Test Invisible Data' to the total number of songs present in the 'Test Invisible Data'. We believe that this is the right measure to evaluate an algorithm because a lower value of precision represents higher degree of error in our recommendation and vice versa which is the right measure to evaluate the performance of an algorithm.

We also use Mean Average Precision to include weightage of the ranks in calculating the precision. Let M be the user-by-items matrix where 1 will appear if the user has listened to the song else 0. Let y denote the ranking given by the algorithm where y(i) = s, indicates that song s has predicted ranking i. Now the precision at each point k from 1 to N, where N indicates the number of top songs to be returned by the algorithm is as follows:

$$P_k(u, y) = \frac{1}{k} \sum_{j=1}^{k} M_{u,y(j)} \text{ , where k =0, 1 …N}$$

Finally, Average precision is calculated as follows:

$$AP(u, y) = \frac{1}{n_u} \sum_{k=1}^{N} P_k(u, y).M_{u,y(k)}$$

Where n_u is smaller of number of positively associated songs and N. Finally averaging over all m users, we get the mean average precision:

$$\text{mAP} = \frac{1}{m} \sum_{u} AP(u, y_{u)} \text{ where } y_u \text{ is the ranking}$$

predicted for use u.

D) Data Set Details

The data set used by us in the project consisted by 10000 users and 105042 songs with an average of around 10 songs per user. The number of triplet records representing the listening history of the user for both training and test data is listed in the table below.

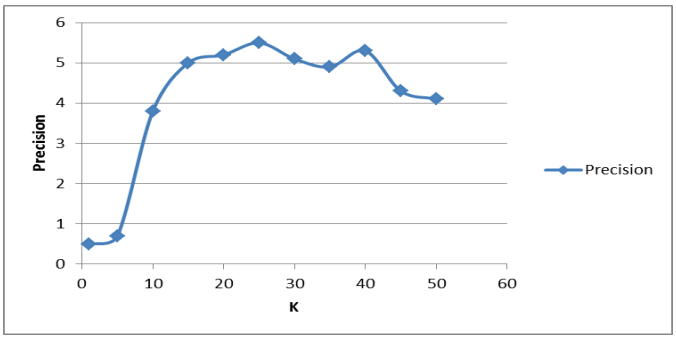
Type of data	Number of records
Train data	479271
Test data	13405

Table I
Dataset Summary

E) Results

In this section we evaluate the different algorithms we have used to build our recommendation system and analyse the results.

1) K-NN – Finding the optimal 'K'

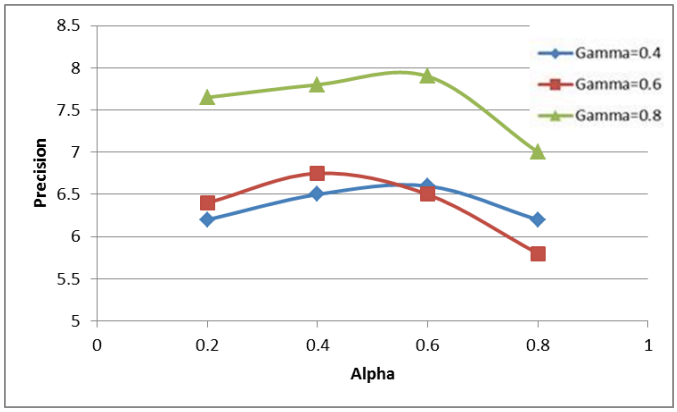


In this experiment we have tested the recommendation model to find the optimal 'K' neighbours value that needs to be used to get the maximum precision for the recommendation of songs to a test user. It was observed that for low values of 'K', the precision of recommendation was low which could be attributed to the fact that recommendation system is more susceptible to the noise in the training model. However with

increase in value of 'K' there was increase in the precision of recommendation. But with larger values of 'K' again low precision in recommendation was observed which could be because of inclusion recommendation from less similar users which again degrades the performance. The graph shown in figure 1 depicts the results of this experiment.

2) User-based CF – Finding the coefficients 'α' and 'ϒ'

In this experiment we try to find the optimal values of the coefficients and see how the precision varies with their values. From the experiments it's found that an optimal value for 'α' is 0.8 and that of 'ϒ' is 0.8. The graph shown in figure 2 depicts the results of this experiment.



3) Performance evaluation of Algorithms with respect to Precision

In this experiment, we have tried to evaluate the three algorithms used to build our recommendation system with respect to their precision. The precision is calculated from the formula,

Precision = # of matches with 'Test Invisible Data'

Total # of entries in 'Test Invisible Data'

Algorithm	Precision
TOP -N	0.8%
KNN	5%
Item Based	6.7%
User Based	7.9%

As expected, Collaborative approaches perform better than K-NN algorithm. This is because in K-NN we recommend the songs to a test user based on the listening history of 'K' neighbours by giving equal weights to each one of them. But in collaborative approaches we use the complete dataset for recommending the songs by calculating the similarity matrix which can be with respect to either users or songs with varying weightage, thus making better recommendations. Among the two CF approaches user-based approach outperforms item-based approach. This is because item similarity matrix is much sparser than user similarity matrix. From the dataset it's observed that on an average a user listens to 47 songs while one song is listened by only 4 users. Therefore, we concluded that item-based CF approaches are more suited for datasets in which number of songs in the dataset is comparable to the number of users so that item similarity matrix isn't sparse. The graph shown in figure 3 depicts the results of this experiment.

V. Conclusion

In this project we developed a music recommendation system using the Million Song Dataset provided by Kaggle. With our recommendation system we were able to achieve a maximum of 7.7% of precision. We realized from this project that building a music recommendation system is very challenging and is more complicated than building a recommendation system for an e-commerce website. This is because various factors like artist of the song, genre of music, mood of the person, lyrics of the song, acoustics of the song etc determine the liking of a song by the user. Hence, when building a music recommendation system all of these factors need to be considered and it's not as simple as saying that if a person buys one item on e-commerce site then he/she might also like another item similar to the one he/she has bought before.

From the experiments it was observed that CF approaches outperformed K-NN approach. It was also seen that user-based CF performed better than item-based CF which due to the difference in the sparseness of their respective similarity matrices. Due to memory constraints of the experiment setup, we had to use a smaller train dataset to train the recommendation system which resulted in low precision of the recommendations made by the system. We didn't use any hybrid recommendation algorithms which otherwise would have improved the performance. As a part of future work, we suggest that other information other than just play count of the songs be used for the recommendation. For example, genre of music, lyrics of the song, artist of song etc could be used in addition to song count for recommending a song to the test user. Also, choosing a large enough dataset to train the recommendation model is very crucial. Small train datasets can hinder the performance of the recommendation system. With respect to the algorithm used to build the recommendation system, it is advisable to use hybrid filtering approaches with ensemble methods on top of it for achieving better performance. Based on the nature of the dataset, right algorithm to build the recommendation system has to be used. The training of the dataset and filtering of the datasets to get the recommendations for a user are both memory and computation intensive. Hence, the recommendations should be run on distributed systems to facilitate fast parallel processing. We believe that though the task of building a highly efficient music recommendation system is very hard buy by making use of some of techniques mentioned above we believe that

we can improve the precision of recommendation of a music recommendation system.

References

[1] http://www.kaggle.com/c/msdchallenge
[2] http://www.pandora.com/
[3] https://www.spotify.com/us/
[4] http://www.beatsmusic.com/
[5] http://www.last.fm/
[6] http://en.wikipedia.org/wiki/Collaborative_filtering
[7] http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[8] Li, Y., Gupta, R., Nagasaki, Y., Zhang, T. "Million Song Dataset Recommendation Project Report". 2012. Unpublished Manuscript.
[9] The million song dataset challenge" Proc. of the 4th International Workshop on Advances in Music Information Research (AdMIRe), 2012
[10] Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE, 7(1), 76-80.
[11] Wang, Jun, Arjen P. De Vries, and Marcel JT Reinders. "Unifying user-based and item-based collaborative filtering approaches by similarity fusion." Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006
[12] Beyer, Kevin, et al. "When is "nearest neighbor" meaningful?" Database Theory—ICDT'99. Springer Berlin Heidelberg, 1999. 217-235.
[13] https://github.ncsu.edu/bkakran/Music-recommendation-System
[14] Chen, Hung-Chen, and Arbee LP Chen. "A music recommendation system based on music data grouping and user interests." Proceedings of the tenth international conference on Information and knowledge management. ACM, 2001.
[15] Benson, Eric A., and Jennifer A. Jacobi. "System and methods for collaborative recommendations." U.S. Patent No. 6,064,980. 16 May 2000.
[16] Konstan, Joseph A., et al. "GroupLens: applying collaborative filtering to Usenet news." Communications of the ACM 40.3 (1997): 77-87.
[17] Resnick, Paul, et al. "GroupLens: an open architecture for collaborative filtering of netnews." Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994.
[18] Sarwar, Badrul, et al. "Analysis of recommendation algorithms for e-commerce." Proceedings of the 2nd ACM conference on Electronic commerce. ACM, 2000.
[19] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." Proceedings of the 10th international conference on World Wide Web. ACM, 2001.
[20] Ali, Kamal, and Wijnand Van Stam. "TiVo: making show recommendations using a distributed collaborative filtering architecture." Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004.