Million song dataset
Music Recommendation System

Mingjun Liu (USC ID: 1321657359)
Xindi Huang (USC ID: 3411804953)
Tian Yang (USC ID: 4979906415)

The Description:

About our Project**:**

There are thousands of songs being published each day, and it's hard for people to find out what they like among such a large number of songs. Nowadays, people may neither don't have much time or energy to create their own playlists nor don't want to go through all music lists to find what they like. Therefore, we are creating a music recommendation system for people who want to discover their taste of music in a short time. We will explore other users' playing history data from Million Songs Dataset [1] and recommend music that the user is most likely to like from users who have similar tastes.

About our Data:

Our dataset is collected from the Million Song Dataset. The Million Song Dataset is a freely available collection of audio features and metadata for a million contemporary popular music tracks. The Million Song Dataset is also a cluster of complementary datasets contributed by the community: so it has subsets focused on different parts, such as cover songs, lyrics, user data etc. The entire dataset is about 300 GB. Considering running time and computer space, we use their subset which was also suggested by the website. This subset has 10,000 songs and Mainly focuses on user listening data. The dataset contains real users - play counts.

New and Innovative about our Project:

Most other music recommendation systems are majorly considering the popularity of songs and singers. Some music recommendation systems even only recommend the newest songs. Compared with those kinds of recommendation systems, in order to maximize the power of the Long Tail, our music recommendation system is going to give a chance to those good music which may not be as popular and not as new. These songs could have a better chance to find their preferred audience. Not only help users to discover more songs, but also help creators and artists. Since we are recommending music to users who would most likely enjoy the music, we could help musicians and artists to find the audience that they deserve. Our recommended system will consider more features to personalize our results such as user, songs, and popularity. We can save People's time and energy while creating their own playlists. And such a playlist is personalized. Not only giving people more choices but also providing a way to add fresh music to their list. Apart from applying and evaluating common techniques, we also tried to combine user-based and item-based collaborative filtering methods which had the best performance in the end.

Pose and Motivate our Hypotheses:

Among all different choices of models in machine learning, we are going to train, test and find the most fitted model to give recommendation results. We are going to use collaborative filtering to exploit similarities in rating behavior among users in determining recommendations and use both item-based and user-based methods to find the similar items that are rated similarly by the same user. Rather than matching similar users, our system will match user's rated items to similar items. Among the models, we were assuming Singular value decomposition methods would have the best performance.

Literature Review:

In the past, content-based models were widely implemented among music recommendation systems to give results of similar songs from current user's music listening history based on music type, singer name, and other features of the songs. Some music recommendations systems give all different users the same outcomes based on the trend of popularity songs.

BellKor recommender system also suggested applying multi-scale modeling to the data, which combined top (global) level, regional effect, with a refined local view. [2] Since this is an algorithm that was originally developed for Netflix movie recommendation, it included time parameters as temporal biases of users. We would modify this algorithm to fit our data without time parameters.

Results:

Evaluation:

Starting with downloaded raw data from the Million Song Dataset, after cleaning, analyzing, and sorting, we split our data into train and test sets. Then we applied the train data set to different methods and models to get results and compare evaluation. There is an additional step for splitting train and test data sets, we split our test data into visible and invisible test sets by having some songs of our users invisible while testing. Thus, we can test all the test users by having corresponding results to compare and evaluate.

We apply mean average precision (mAP for later) as the metric for evaluation, which is the mean over the average precision scores for each user. This method is a good metric to evaluate predictions from our recommender because it includes weightage of ranking and emphasizes the top recommendations. [3] For a total number of N recommended songs to a user, it summarizes the precision at each point k, which is the proportion of correct recommendations within top-k of the predicted ranking. Below are the steps for evaluation.

Assume $M_{(u,s)}$ is the User-by-items matrix where 1 means the user has listened to the song in the hidden set, and 0 means has not and $y(i) = s$ means y is the list of recommendations to the user, song s has the ith position in the ranking. For the songs recommended, we find

$$p_k(u, y) = \frac{1}{k}\sum_{j=1}^{k} M_{u,y(j)}, where\ k = 1, 2, \ldots N$$

which is the precision at each point k from 1 to N. For each user, we calculate the average precision at each recall point

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

where nu is the smaller number between N and positive associated songs of the user. Finally, mAP is taking the mean of AP over all the users. We apply this metric because it can tell us the true positives numbers proportional to the total number of predictions, and a large false positives number means seldom of our recommendation works, which potentially leads to bad users' experience.

Methods:

In our project, we used 5 different methods to implement the recommendation system: popularity-based method, K-Nearest Neighbors based method, Collaborative Filtering techniques, Mixed user-based and item-based CF, and Matrix Factorization.

A. Popularity-based:

The first and the simplest baseline method we used is based on popularity of songs. The recommendation system will recommend the most popular songs to users. The popularity was calculated by summing all of the play counts of one song among all the users in the dataset. However, the recommendation result produced by this method was not personalized.

B. K-Nearest Neighbors-based:

The second method we implemented was based on the K-Nearest Neighbors (KNN) algorithm. For each test user, we defined their play count for each song in the dataset as their feature vectors. Then, calculate the cosine similarities between the test user and other users to find K closest neighbors. After that, among those K neighbors, we explored their listening history and found N most listened songs and recommended them to the test user.

One important parameter in this method is the number K, and finding the optimal K will help us get the maximum precision of this method. This is a chart showing the number K v.s. mean average precision. At first the precision would get greater as we increase K. However, it would start decreasing after a number. In our project, we found that K=7 will give the best result.

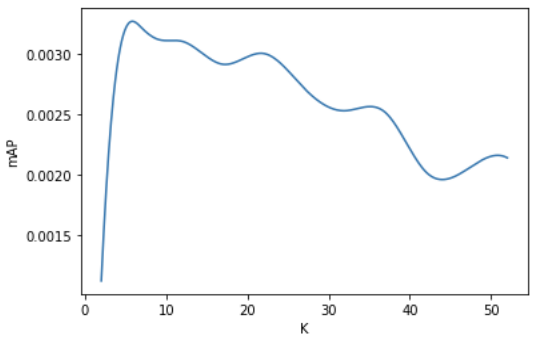

Figure 1 Finding Optimal K

C. Item-based/User-based Collaborative Filtering:

The third method we implemented was Collaborative Filtering technique. For this method, we explored similarities in rating behavior among users. We used both user-based and item-based Collaborative Filtering in our project.

As for user-based, users who have similar listening histories and ratings are defined as similar users. For each test user, we found N other users whose ratings are similar to him. This was done by calculating the Pearson similarity. [4]

$$w_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_i})^2} \cdot \sqrt{\sum_{u \in U}(r_{u,j} - \overline{r_j})^2}}$$

Then estimated his ratings based on those users by weighted average.

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} |w_{i,n}|}$$

As for item-based, it's similar to user-based but the key idea here is similar items will be rated similarly by the same user.

D. Mixed Model:

Since item-based and user-based didn't perform well as expected, we also tried to combine these two models linearly. Score produced by item-based method $s_{u,m}^{item}$, and user-based method $s_{u,m}^{user}$ should be normalized first due to different scales. Then, two linear weights δ and $(1 - \delta)$ are assigned to them, new score for song m towards user m can be calculated as:

$$s_{u,m}^{mixed} = \delta \cdot normalized\ s_{u,m}^{item} + (1 - \delta) \cdot normalized\ s_{u,m}^{user}$$

After tuning the parameter δ, our mixed model performed better than both item-based CF and user-based CF.

E. Matrix Factorization:

The last method we implemented was Matrix Factorization. Since the user-item matrix was extremely sparse for our dataset, we used Singular Value Decomposition (SVD) which was able to remove unrepresentative or insignificant users or items thus reduced the dimensionality of the matrix. The idea here is approximating the rating matrix R by $R \approx Q \cdot P^T$ where $Q \cdot P^T$ has less entries than R but summarizes the rating features in R. [5] It's important to include bias terms and generalization in the data, we would need to determine latent factor in this method.

Bias term	Mean average precision
user	2.8793825222396653e-05
song	2.9334950763522196e-05
both	2.8793825222396653e-05

Above shows the mean average precision when we applied different bias terms into the matrix factorization method for training data. It suggests that applying bias to songs only will have a better result on testing data.

Analysis:

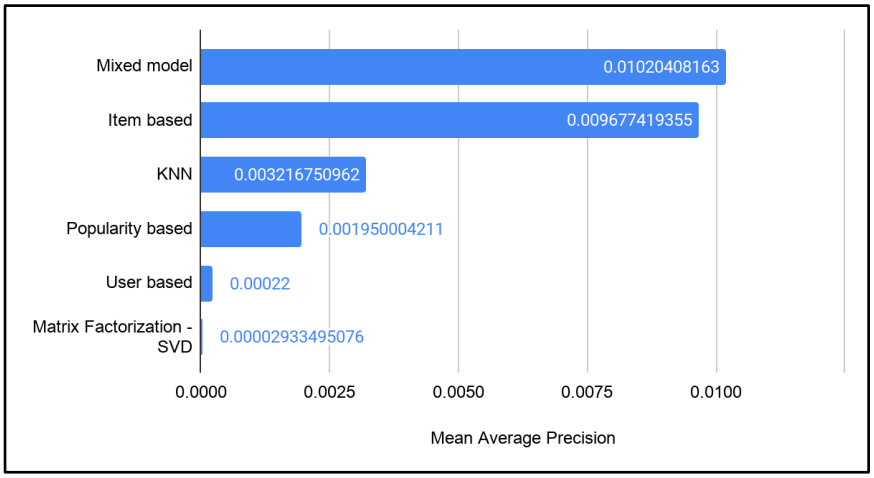

Figure 2 mAP Results of Different Methods

Mixed model performed best in our project. Item based collaborative filters also have an outstanding precision compared to the other methods. This is because the ratio of songs per user is larger than users per song, therefore the number of features for items is larger, and easier to find perfect similar items. Under this situation, it recommends applying an item-based method when the number of the user is small. The runtime of the training process also limits us to reduce the size of the user sample, which results in hardship to find perfect matching similar users. Singular value decomposition (SVD) had the worst result in our project. Although matrix factorization has compelling advantages in the recommendation system field, our dataset is too sparse which causes the objective function of SVD cannot converge to a global optimum.

Conclusion

In this project, we apply recommendation methods to different datasets, choosing the corresponding parameters, and learn evaluation methods for recommendation systems. Although we assumed Singular value decomposition would have the best performance, it turns out that it has the worst result. We also introduced the mixed method that combines user-based and item-based CF for finding optimal solutions. There exists a cold start problem for recommendation systems, and it can be solved by the popularity and matrix factorization methods. However, these are not the optimal solution we found in the project. Our future work will focus on optimizing the solution for the user-training process and looking for feature clustering that helps solving the cold start problem.

References:

[1] 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.

[2] Koren, Yehuda. The BellKor Solution to the Netflix Grand Prize. August 2009. https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

[3] National Institute of Standards and Technology. TREC-2004 common evaluation measures. Available online at: http://trec.nist.gov/pubs/trec14/appendices/CE.MEASURES05.pdf (retrieved on August 27, 2007).

[4] 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

[5] Cremonesi, Paolo, Yehuda Koren, and Roberto Tur- rin. ”Performance of recommender algorithms on top- n recommendation tasks.”Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010