Music Recommendation Using Collaborative Filtering

Vuong Khuat
Department of Computer Science, Earlham College

Introduction

In the last 25 years, the use of recommender systems has expanded rapidly. These systems, designed to produce recommendations that match users' interests, are incorporated into areas such as news, entertainment, and research. One common technique used by recommender systems is collaborative filtering, which produces recommendations for a user based on choices of users with similar preferences. This work describes a recommender system for music tracks using collaborative filtering.

Collaborative filtering is widely used; however, the method does not scale well with large datasets. It is important to build a recommender system using collaborative filtering that not only is accurate but also runs efficiently. In this project, I attempt to build a recommendation systems for music tracks using collaborative filtering with two optimization methods, dataset rescaling and automatic halting. These methods will be built and tested with different parameters to address the performance issue while retaining an acceptable accuracy.

Background

1. Collaborative Filtering

Collaborative filtering provides recommendations for a target user by estimating the utility of different items based on the habits or ratings of other users. The advantage of this technique is that the behavior of a user can generally be predicted from users who are similar. For example, suppose we have a user-song table as shown below, where 1 denotes a user has listened to a song, and 0 if they have not:

$$\begin{array}{cccc} & u_a & u_2 & u_3 \\ s_1 & 1 & 1 & 1 \\ s_2 & 1 & 1 & 1 \\ s_3 & 1 & 1 & 1 \\ s_4 & 1 & 1 & 1 \\ s_5 & 0 & 1 & 1 \end{array}$$

The algorithm in this case may determine that u_a, u_2, and u_3 are similar (since they all listen to s_1, s_2, s_3, s_4), and recommend track s_5, which has been listened to by both u_2 and u_3.

2. Types of Ratings

Determining the similarity of different users requires recommender systems to attain a numerical value for every user-item pair which represents the user's rating of the item. There are two types of user ratings that can be used by recommender systems, *explicit rating* and *implicit rating*.

Explicit rating	Implicit rating
• Users explicitly provides feedback for different items on a numerical scale. • i.e. from 1 to 5 where 5 means "strongly like" and 1 means "strongly dislike") • Ratings are provided directly by users so is generally considered a good rating system. • However, it is not practical for users to provide ratings for every item, so can make use of only rated items or derive implicit feedback from the rest.	• Ratings are not provided by users, but are predicted based on users' activity. • For instance, if someone has not provided a rating for a certain song but has listened to it more than 20 times, she will likely give the song a high rating.

Acknowledgements

This project is a part of the Senior Capstone Experience of the Department of Computer Science at Earlham College. I would like to thank Dr. David Barbella, Dr. Ajit Chavan, Dr. Xunfei Jiang, and Dr. Charles Peck of Earlham College for their support in this project.

Methodology

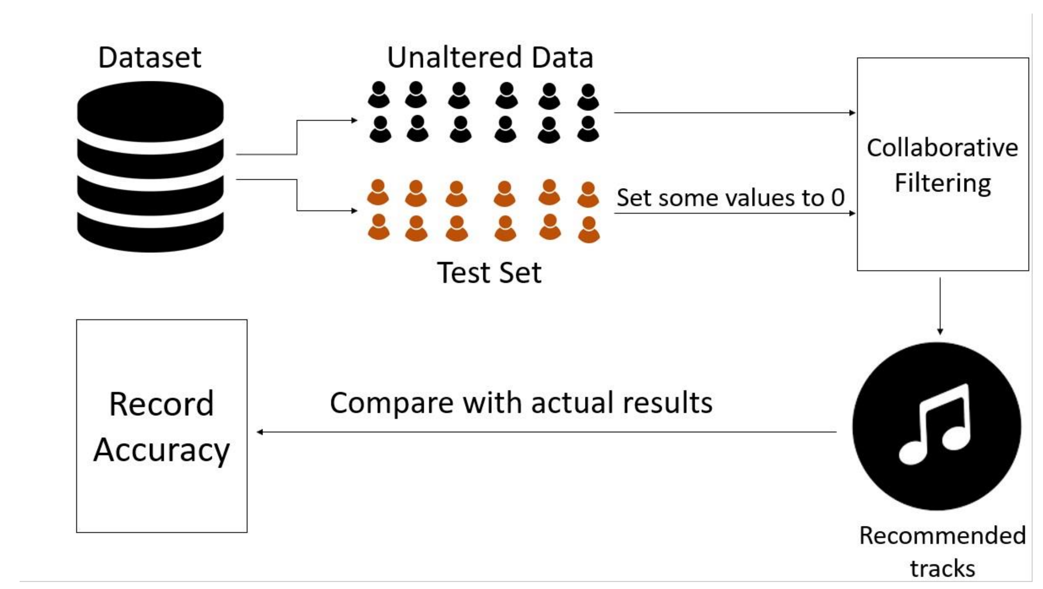


Figure1. Overall framework of the project

1. Processing the Data

In this project, the data will be collected from the Echo Nest Taste Profile Subset, which provides play counts of more than 300,000 users for approximately 1 million songs. Traditionally, the data is represented as a 2-dimensional matrix, but this is not space-efficient. Instead, we represents the dataset as a hash table that maps each user to the set of songs they listened to. This works because in practice the most active users of large commercial websites consume less than 1% of the contents.

2. Implementing the Collaborative Filtering Algorithm

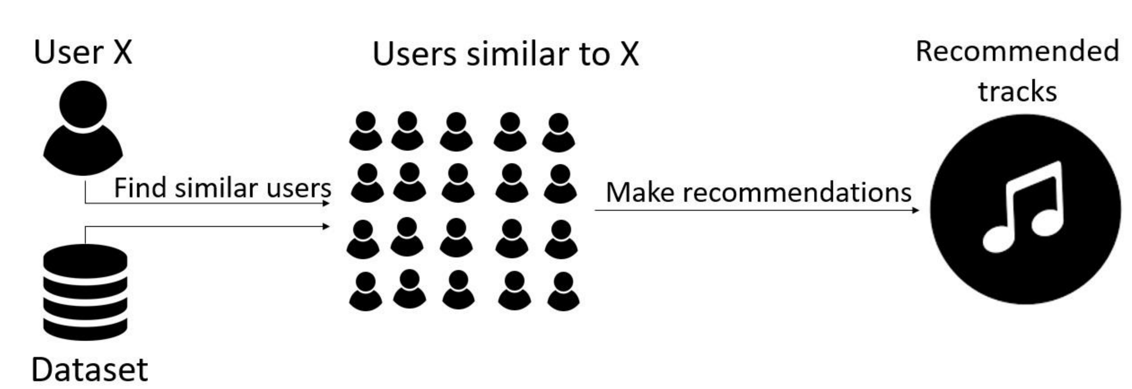


Figure 2. Overall framework of the project

The collaborative filtering algorithm will be implemented as a function f. The function takes in the dataset d, a user u_a, a positive integer N and returns an list that contains N songs that best match u_a's interest. This is done in two steps.

First, f determines the users who are similar to u_a. This can be done using a iterative process: for each user u_i that is not u_a, if u_i and u_a are similar, then add u_i to the list of similar users.

Having produced a list of similar users, recommendations can then be generated for u_a from music tracks that the a specific proportion of similar users have listened to but u_a has not.

3. Determining the Similarity Between Two Users

To determine the similarity of users, this work consider users as m-dimensional vectors, where m is the total number of songs in the dataset. With this representation, the similarity of two users, u_i and u_j can be determined by the cosine of the angle between the corresponding vectors:

$$cos(\vec{u}_i, \vec{u}_j) = \frac{\vec{u}_i \cdot \vec{u}_j}{||\vec{u}_i||||\vec{u}_j||}$$

We define a parameter θ to indicate the largest value the angle between u_i and u_j can have for them to be considered similar. In evaluation, we test the model to determine the best value for θ.

Testing and Evaluation

Given a user u_a, we retain k components of the vector u_a, where k is a positive integer less than m, and set the remaining m – k components to 0. We can assume that u_a has the form

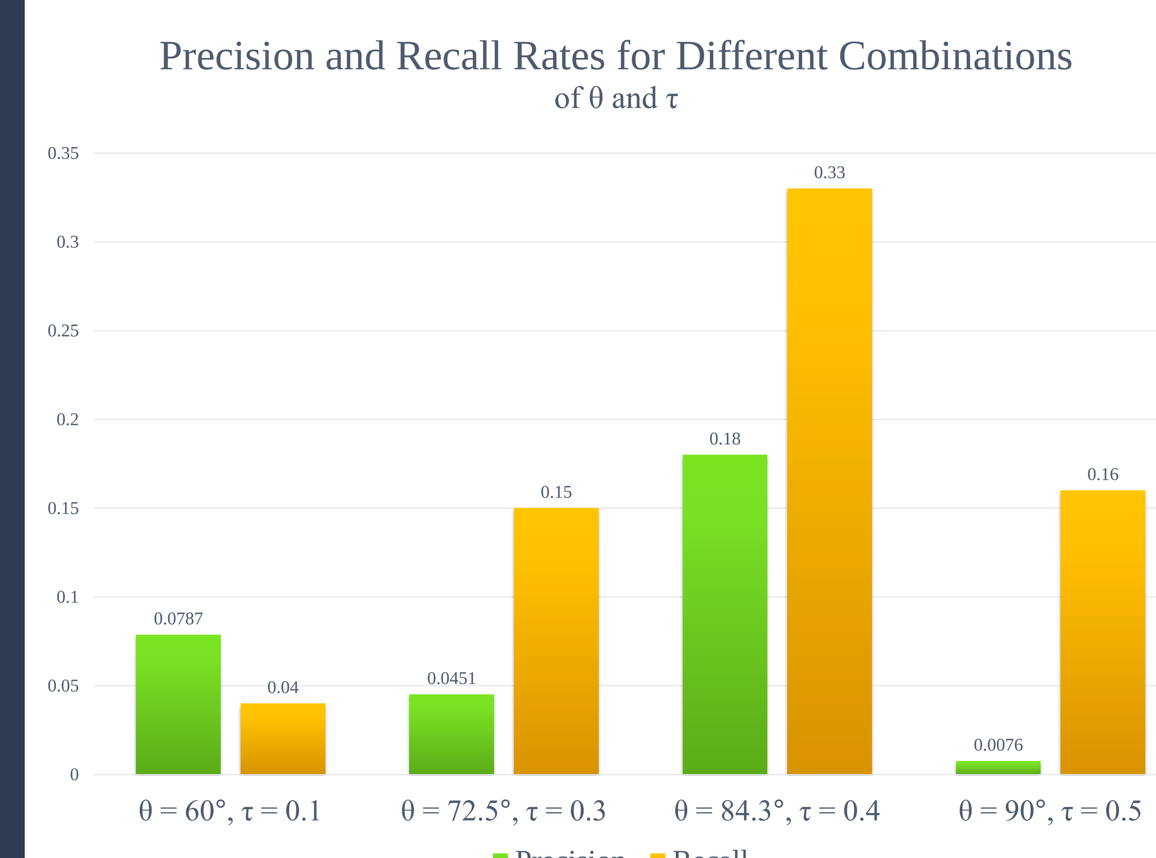
$$\begin{array}{cc} s_1 & \text{real value} \\ s_2 & \text{real value} \\ \vdots & \vdots \\ s_k & \text{real value} \\ s_{k+1} & 0 \\ s_{k+2} & 0 \\ \vdots & \vdots \\ s_m & 0 \end{array}$$

Then, we apply the collaborative filtering algorithm to generate recommendations for u_a. To prevent the model from producing too many recommendations, we set the maximum number of recommendations to be 10k, and the songs chosen are the most frequent songs among the list of similar users. The components that were set to be 0 are now compared with the corresponding real data.

The model is assessed based on the runtime as well as the following measures:

- **Precision:** the number of songs which were hidden that are in the list of recommendations out of all recommended songs, and
- **Recall:** the number of songs which were hidden that are in the list of recommendations out of all hidden songs.

Let τ represent the proportion of similar users that must have listened to a song for that song to be recommended, and let θ be the largest value the angle between u_i and u_j can have for them to be considered similar. The following graph shows precision and recall rates, respectively, for different combinations of τ and θ.



Methods to Improve Runtime

In this work we implement two methods that are used to reduce the runtime.

a. **Randomly select a subset of the dataset.** This method select a fixed-size subset of the dataset, which is generated at random. The subset-size parameter is determined from experiments to see whether our model can run within a reasonable amount of time without decreasing the accuracy rate by a significant amount.
b. **Halt after finding enough similar users.** The execution is stopped once the number of similar users found is sufficient to generate recommendations.

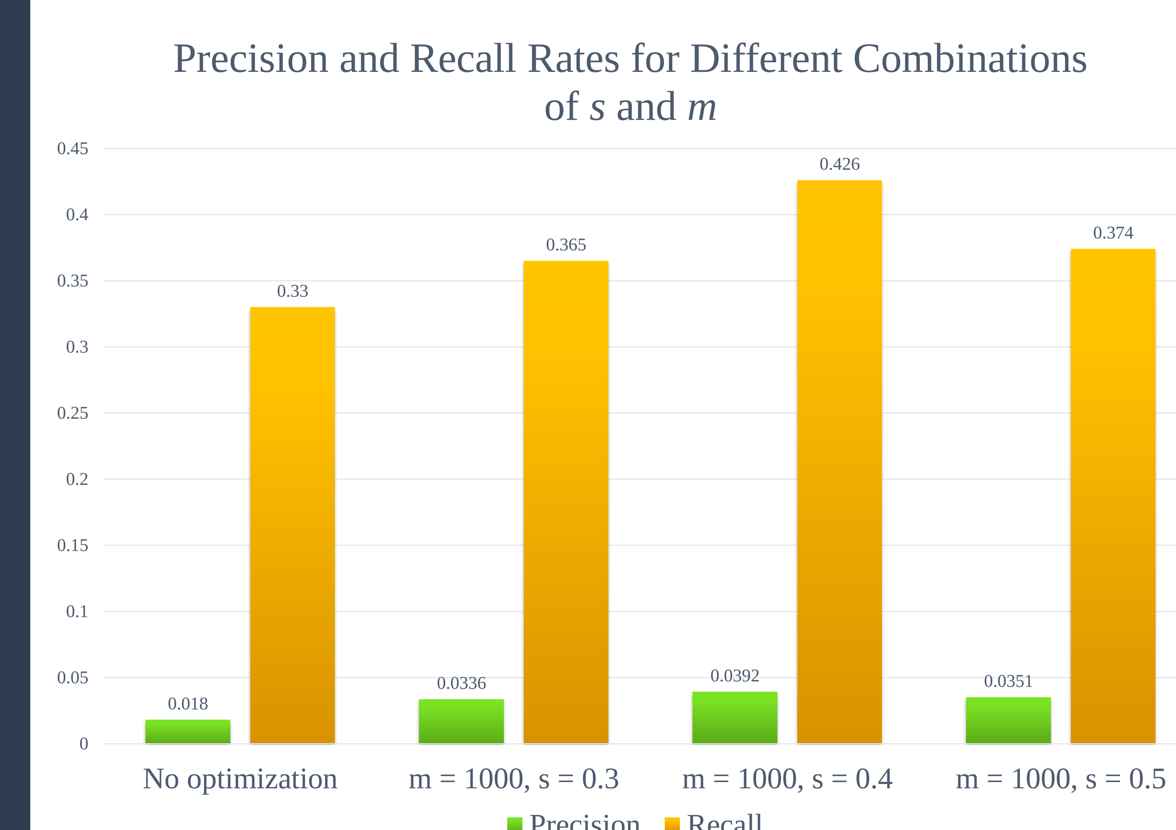
Experiment Results and Conclusion

Denote s as the proportion of the generated subset compared to the full dataset, and m as the number of similar users threshold to halt. We use θ = 84.3° and τ = 0.4 as these values give the best recall rate in the previous step. The execution time of different values for s and m are shown in table 3.

s \ m	500	1000	2500	5000
0.3	1.8247	1.938	2.001	2.0460
0.4	2.194	2.5218	2.590	2.628
0.5	2.760	3.0783	3.208	3.297

Table 3. Execution time (in seconds) for different combinations of s and m

We also record the precision and recall rates for these test runs. The results are documented in the following graph.



The average runtime before deploying these methods is 5.81 seconds, whereas the runtimes documented in table 3 range from 1.83 to 3.3 seconds. Lower values of s and m further reduces the runtime. Furthermore, the precision and recall rates attained using these methods are both slightly higher than the corresponding values in tables 1 and 2. The results of these experiments show that the runtime-improvement methods both reduce the overall runtime and retain the accuracy of the model.

From the results of this project, for related future work, we would like to explore the hypothesis that the best accuracy of this model is attained after finding a certain number of similar users. We would also want to explore other methods which manipulate the dataset to reduce the runtime. Such methods would need to consider properties of the dataset so that the collaborative filtering function can make less comparison without losing accuracy.

References

- Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2000-10-17. Analysis of recommendation algorithms for e-commerce. Proceedings of the 2nd ACM conference on Electronic commerce (2000-10-17), 158–167.
- Thierry Bertin-Mahieux, Daniel PWEllis, Brian Whitman, and Paul Lamere. 2011. The million song dataset. (2011).