Netflix Recommendation System Final Report

Zhengyang Zhang
Schaefer School of Engineering
Stevens Institute of Technology
Hoboken, USA
zzhan67@stevens.edu

Chenxu Yi
Schaefer School of Engineering
Stevens Institute of Technology
Hoboken, USA
cyi2@stevens.edu

Zhaonian Zhu
Schaefer School of Engineering
Stevens Institute of Technology
Hoboken, USA
zzhu32@stevens.edu

I. Introduction

Today we live in a world where recommendation system applied in many aspects of our lives. When we go on Amazon, Amazon automatically recommend related product to you because you purchased certain item, that's recommendation. When you go on Spotify, they automatically recommend playlist/songs you may like based on your past listening history, that's also recommendation. As we can see, recommendation system is everywhere in our lives, it helps create more personalized experiences for us as users.

In this project, we want to use Netflix Price dataset on Kaggle to build a complete set of important recommendation system for Netflix.

Through this project, we want to create three type of recommendation system: Rank-Based Recommendation System, User-User Based Recommendation System and Content-Based Recommendation and connect them together to become a whole recommendation system. In the end, we also want to test how the number of latent features is going to affect the accuracy of the system.

II. Data Description

We selected data from Netflix Price Dataset on Kaggle. The dataset link is below:

https://www.kaggle.com/netflix-inc/netflix-prize-data

The data set consists of several data files:

- combined_data_1.txt: data includes UserID, MovieID and Rating part1
- combined_data_2.txt: data includes UserID, MovieID and Rating part2
- combined_data_3.txt: data includes UserID, MovieID and Rating part3
- combined_data_4.txt: data includes UserID, MovieID and Rating part4
- movie_titles.csv: data includes MovieID, Production Year and Movie Title

III. Exploratory Analysis

If we combine the first four dataset, there are total 26847522 ratings in which different users rated different movies. And there are total 17769 different movies in this dataset.

And we count the number of interaction each unique user have and we have the following histogram:

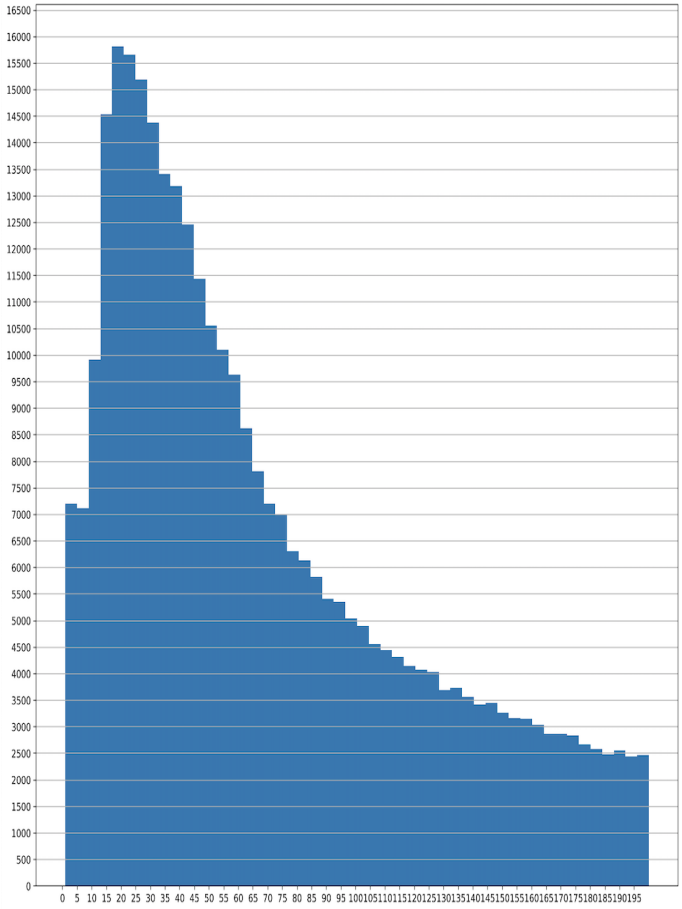


Fig. 1. distribution for number of iterations

Descriptive Statistics

- count 480189.000000
- mean 209.251997
- std 302.339155
- min 1.000000
- 25% 39.000000
- 50% 96.000000
- 75% 259.000000
- max 17653.000000

As we can see from the graph and descriptive statistics above, we can see that there are users who almost interacted and rated every movie on Netflix, and there are also users who only interacted with as minimum as 1 movie. But the average number of interaction is 209.

IV. Recommendation System

Below we're going to give some brief introduction on each recommendation system and its core algorithm and ideas as well as its pros and cons when comes to different user types.

A. Rank Based Recommendation

Introduction

Thanks to the Internet, the freedom of information ensures us to achieve anything if we want. However, everyone only has 24 hours a day, so the remaining time to obtain information is limited after regular working and studying time. Also, laziness is the nature of humans. Based on these problems, we have an intelligent recommendation system that gives you advice after product use. But what is the most efficient technique to recommend when we have many new users who do not use our products before? The answer is Rank Based Recommendations.

Rank-Based Recommendation is the most fundamental recommendation system. Because of the Rank-Based, users could know what is popular currently. For example, Netflix always displays a row called "Popular on Netflix," which will let users know the popular series recently. Like Netflix, our Rank-Based Recommendation could count the views of movies and recommend the most viewed movies to users. If we set the recommend movies amount as 10, users will get the Top 10 viewed movie's names.

Pros and Cons

Pros

- It is easy and fast to reach the goal. You only need few lines of code to find out the Top N movies.
- It is practical and could be used in a wide range. Users depend on this recommendation system a lot no matter they are new or old users because everybody wants to catch up the trends and the popular movies are great too

Cons

- The recommendation result is too vague. People may not satisfy with the preliminary result; they want more recommendations related to their interests.
- The product which only uses Rank-Based recommendation will not develop sustainingly. The Rank-Based recommendation always gives a famous or popular result which people are familiar with it. As a result, the product does not have core values.

Fig. 2. Rank-based recommendation

B. User-User Based Recommendation

Introduction

Collaborative Filtering is the most common technique used when it comes to building intelligent recommender systems that can learn to give better recommendations as more information about users is collected.

In the project, we firstly build a matrix with UserID as rows and MovieID on the columns with 1 values where a user interacted with a movie and a 0 otherwise (Use Rating as a metric). Next, without knowing anything about items and users themselves, we think two users are similar when they give the similar item similar ratings.

We find measure the similarity based on cosine similarity. Cosine similarity is a measure of similarity between two non-zero vectors. In the project, the vectors are the users.

Pros and Cons

Pros

- We don't need domain knowledge because the embeddings are automatically learned.
- The model can help users discover new interests.
- The system needs only the feedback matrix to train a matrix factorization model.

Cons

- Cannot handle fresh items.
- Hard to include side features for query/item.

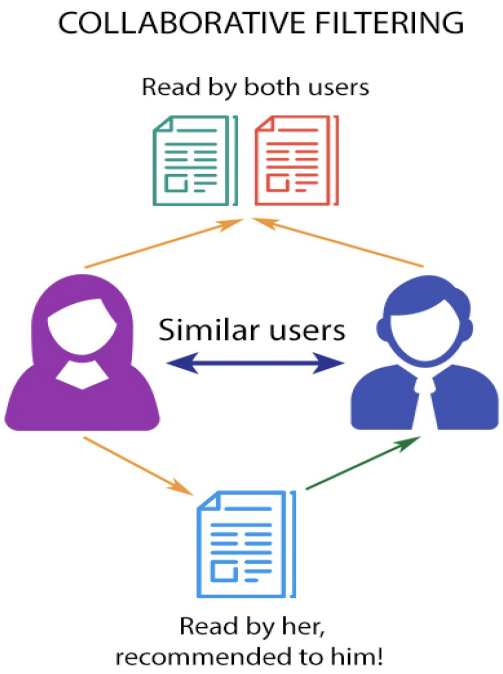


Fig. 3. collaborative filtering recommendation

$$Cosine\ Similarity: Sim(u_i, u_k) = \frac{r_i \cdot r_k}{|r_i||r_k|} = \frac{\sum_{j=1}^{m} r_{ij} r_{kj}}{\sqrt{\sum_{j=1}^{m} r_{ij}^2 \sum_{j=1}^{m} r_{kj}^2}}$$

Fig. 4. cosine similarity

C. Content Based Recommendation

Introduction

Content-based recommendation systems recommend items to a user by using the similarity of items. This recommender system recommends products or items based on their description or features. It identifies the similarity between the products based on their descriptions. It also considers the user's previous history in order to recommend a similar product.

In the project, we did a content based recommendation on Netflix movies base on the movie titles. We utilized natural language processing techniques such as word tokenization and lemmatisation to help find words that are similar and share the same root.

The basic idea on what we implemented is we found the top 10 most common word in all movies titles that a certain user have watched, and we count the number of occurrences of those word in other movies that this user haven't watched. And we recommend those movie to users that has highest number of occurrence of common word in it.

Pros and Cons

Pros

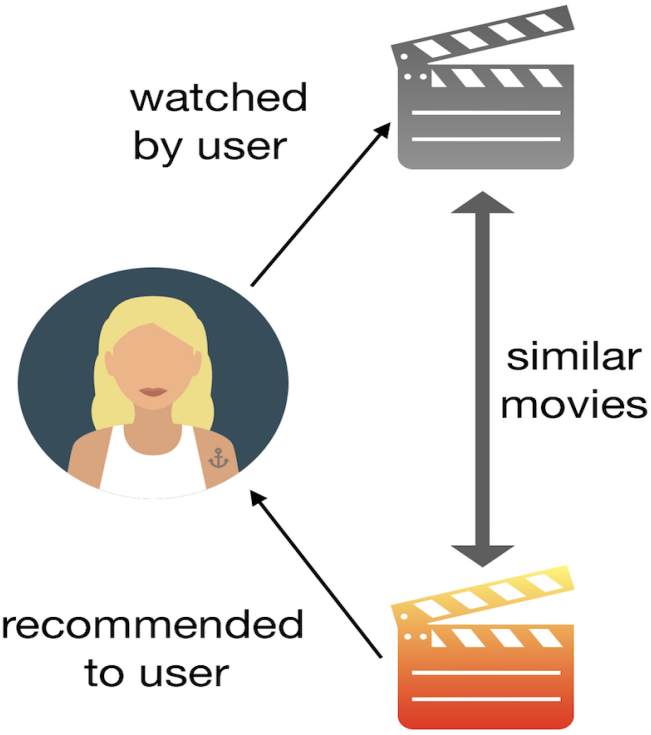


Fig. 5. content-based recommendation

- The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
- The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

Cons

- Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.
- The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

V. Matrix Factorization

In this section, we performed Matrix Factorization to see how the number of latent features(latent features are 'hidden' features to distinguish them from observed features. Latent features are computed from observed features using matrix factorization) could affect the accuracy of the recommender system.

We utilized the idea of SVD (Singular Value Decomposition) and reconstruct the matrix using truncated SVD with number of latent features and use the reconstructed matrix to compute the accuracy of the system.

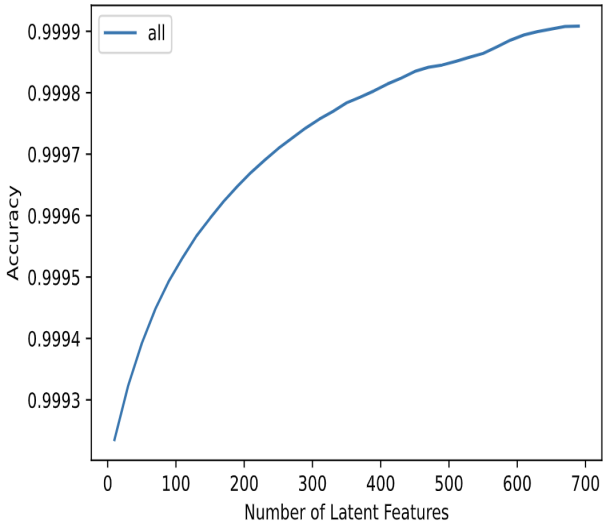


Fig. 6. Number of latent feature vs accuracy full data

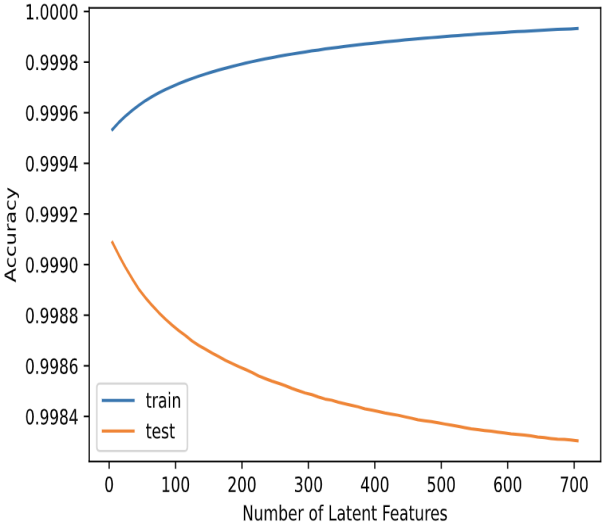


Fig. 7. Number of latent feature vs accuracy train test

If we only look at the figure 3, we would all believe that more latent feature would give us higher accuracy. However, if we look at figure 4, we would notice that the testing set is suggesting less latent feature would give us higher accuracy.

The explanation for this is that with increasing latent features causes overfitting during training. And the model accuracy could be this high because this is mostly a sparse matrix, therefore we might not need to use very many latent features to correctly reproduce the original matrix.

We believe we should not implement a recommendation system just yet solely using SVD as the training and testing sample is still quite small. Since we only have overlap of a few users and some movies between training and testing, I cannot detrmine with high certainty that the SVD recommendations work well in this case.

For example we could conduct an A/B test to see the effectiveness of the matrix recommendation system in comparison to a different recommendation system like rank-based, collaborative filtering etc.

The null-hypothesis in this case would be that there is no difference between the matrix recommendation system and the rank based recommendation system in terms of the number of views on the recommended movies. This could also be potentially expanded to the time spent on the movie when clicked through as well. This would give a more accurate idea of how much that user likes that movie.

The A/B test would show one group only the rank based recommendations while the B group would see only the matrix recommendations. We could split the users by cookie based diversion, so that an equal number of cookies are split between A and B groups. This would be the invariant metric.

We would have to check each evaluation metric for statistical significance. If not each evaluation metric is significant we would have to make a decision about whether we still want to deploy our new system.

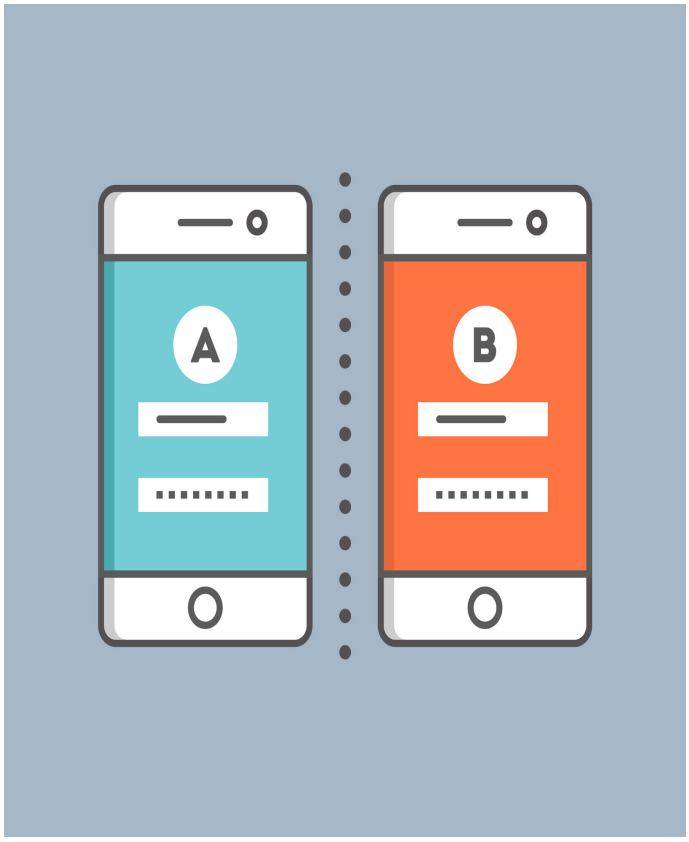
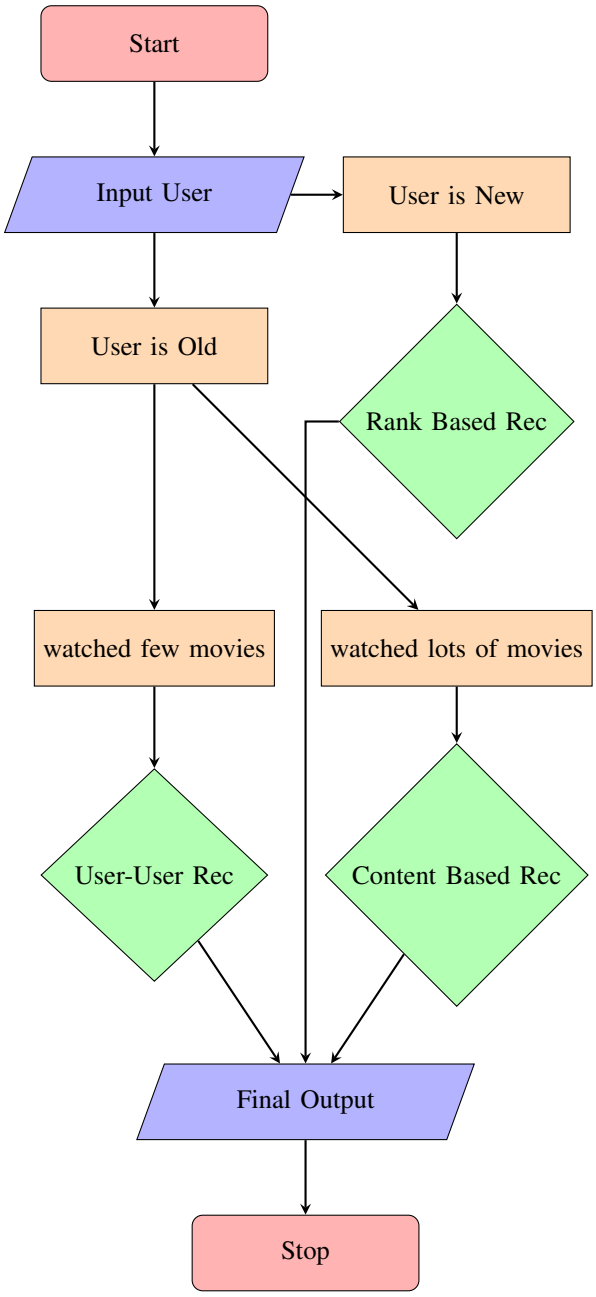


Fig. 8. A/B testing

VI. System Design

With all the system we mentioned in this report, we want to combine them into a complete of system which can determine how to recommend to different users. The graph below is a flow chart that indicate how different user is getting recommended movies through different type of recommendation system.



VII. Conclusion

We build a whole recommendation system that included rank based recommendation system, user-user based recommendation, also known as collaborative filtering and content based recommendation. However, due to the size of the data and the nature of the recommendation system, we're unable to evaluate the accuracy of the system. The system could be better if we consider the rating as an actual factor when recommending movies to users, instead of only recommending base on the user's interaction with movies. And another thing, in the system design, we have no idea how much movies are watch is considered "a lot" and "few", and this is worth further investigation on this topic.