Movie Recommendation System

1. Introduction

Currently, there are around 500,000 movie titles listed on IMDb. As thousands of new movie titles are released each year around the globe, it can be fairly difficult to navigate through the sea of information and discover the right movie to watch. I would like to build a movie recommender system using publicly available movie and movie-review dataset to help find movies worth watching.

2. Literature Review

Movie Recommender System [1]

This project aims to create a scalable recommender system for movies. First, different models were prepared and evaluated on a smaller dataset. Then the model was later implemented on the larger dataset. The authors tried Nearest Neighbors Method and Latent Factor Methods for Collaborative Filtering, and Cosine Similarities based method for Content-based Filtering. In conclusion, User-based collaborated filtering performed better compared to the other methods.

A User-Centric Diversity by Design Recommender System for the Movie Application Domain [2]

This project discusses how accuracy-only focused recommender systems can negatively impact user experiences. For example, users would be recommended with a set of very similar recommendations. The authors aim to enhance the user experience by introducing a new framework using different diversity measures in addition to the traditional accuracy-based recommender system.

The authors introduce a framework to control recommendation diversification with three different diversity properties: variety, balance, and disparity. The diversification procedure is incorporated prior to the extraction of the items, and hence outputting more diverse set of recommendation compared to the previous method of diversifying after extracting the recommendations. Although, an actual study on how such diversification enhances user satisfaction is yet to be done, and room for improvement is still present with the optimization, genuine framework for the diverse recommendation was very interesting.

An Improved Collaborative Filtering Based Recommender System using Bat Algorithm [3]

This project examines how Swarm Intelligent techniques improve Recommender System. The authors implement Bat Algorithm to improve Conventional Collaborative Filtering. Bat Algorithm is based on how micro bats use SONAR to locate prey and circumvent barriers. Functions replicating bats echolocation behavior are coded, locating the optimal solution for recommender problem.

The Jester dataset was used for the experiment. Bat Algorithm outperformed Artificial Bee Colony in terms of MAE and F1-Score by 6.9%.

An effective collaborative movie recommender system with cuckoo search [4]

This project discusses a novel recommender system with k-means clustering and cuckoo searching algorithm. Cuckoo search is an optimization algorithm, where the algorithm was inspired by cuckoo birds laying their eggs in other host birds. Each egg in a nest represents a solution and a cuckoo egg represents a new solution. The algorithm aims to produce a better solution by replacing sub-optimal eggs with a better egg as the generation goes on.

The authors worked with the MovieLense dataset and found adding cuckoo search algorithm improves traditional collaborative filtering. However, the model has some initialization issues, where model efficiency suffers when the initial partition does not turn out to work well.

3. Dataset

MovieLens 1M Dataset – available: https://grouplens.org/datasets/movielens/1m/

ratings.dat – 1,000,209 observations

- userId (string) – unique ids for individual users
- movieId (string) – unique ids for individual movies
- ratings
- timestamp (integer)

movies.dat – 3,883 observations

- movieId (string) – unique ids for individual movies
- title (string) – movie title
- genres (string array)– pipe-separated list of genres

ratings.dat:

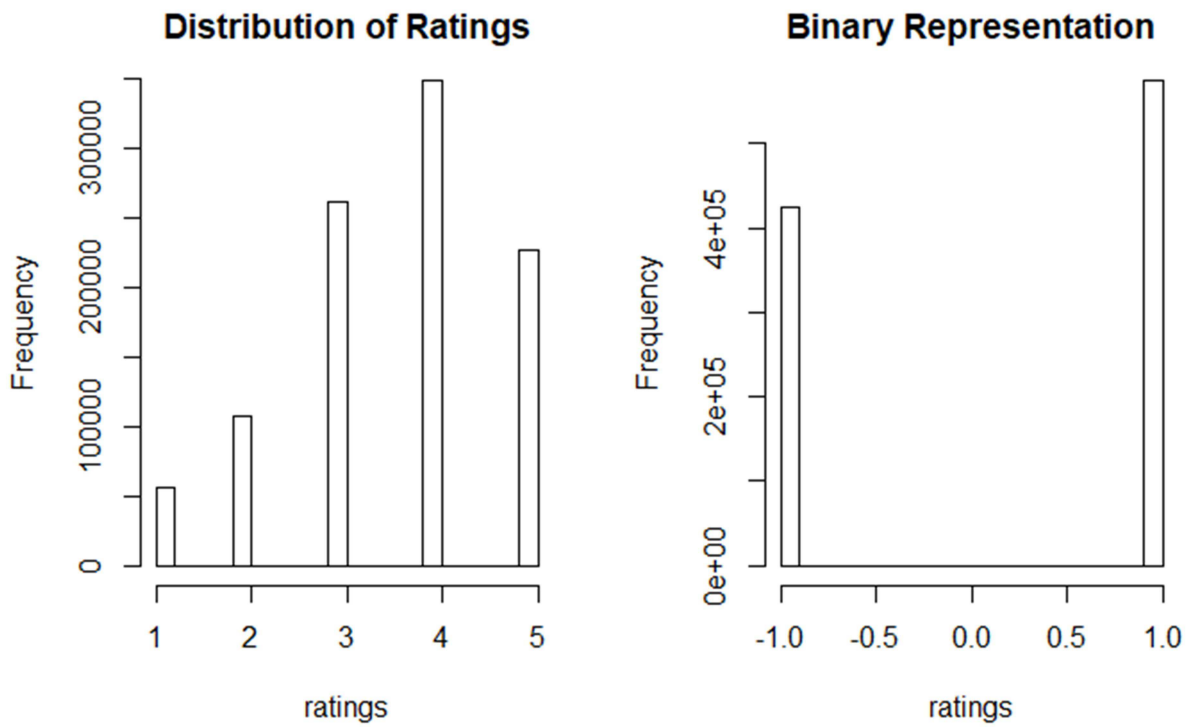

Figure 3.1 - Distribution of ratings

Ratings are distributed with a mean of 3.58 and a median of 4. I will instead work with binary representation of ratings where 1 representing a positive rating of higher than 3, and -1 representing negative ratings of 3 and lower. With the binary representation of ratings, 57.5% of ratings were positive, and 42.5% were negative. There were no missing values regarding ratings.

Within the ratings data, there were ratings from 6,040 unique users and 3,706 unique movies. Each user has rated at least 20 movies, but there was no guarantee that all the movies were rated by more than one user. Every combination of userId and movieId were unique, and ratings were made exactly once and each rating was unique.

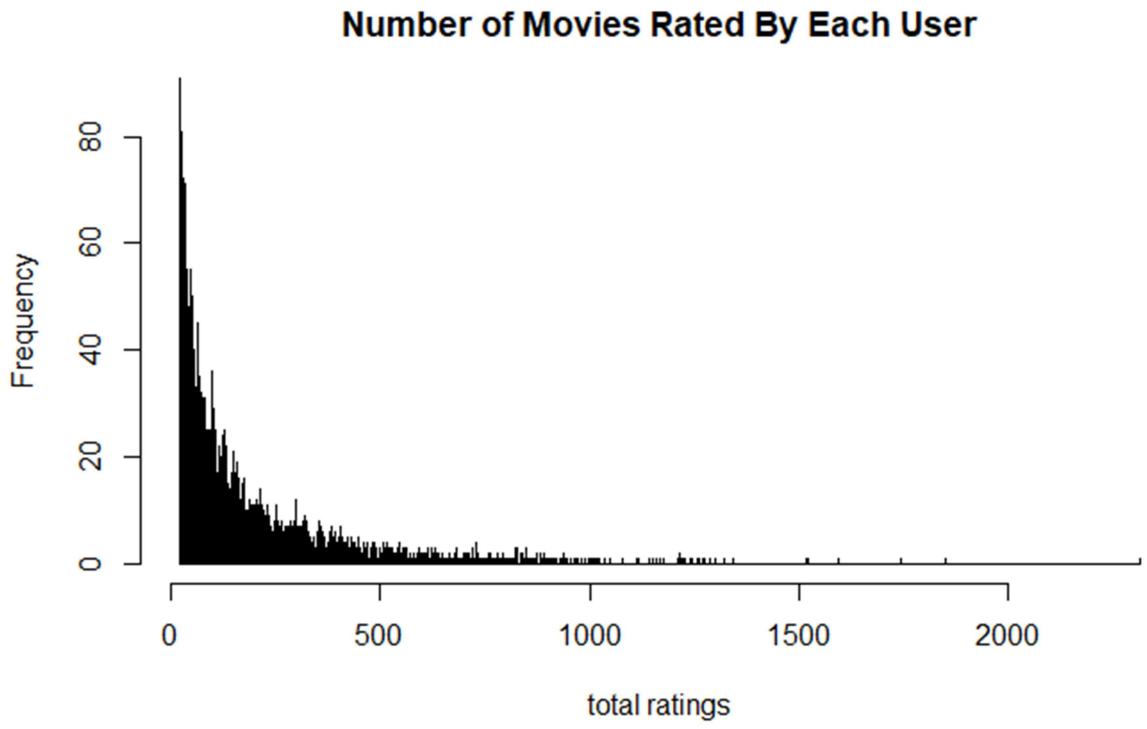

Figure 3.2 - Total number of movies rated

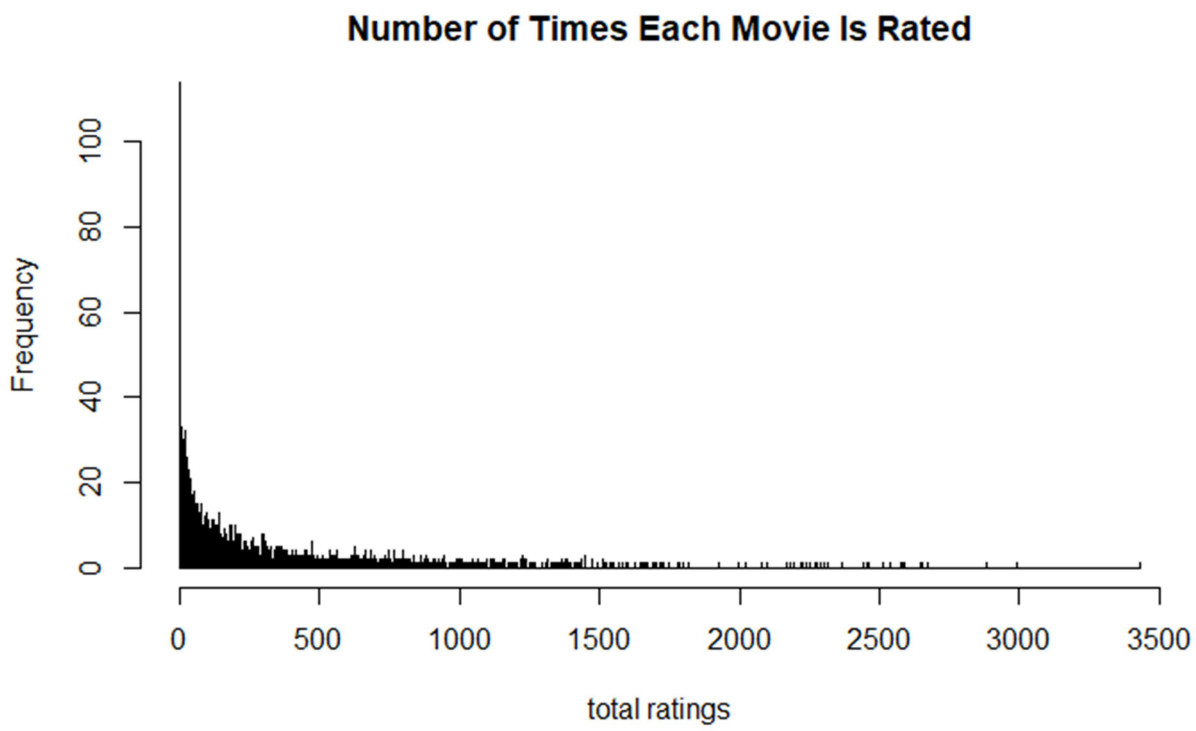

Figure 3.3 - Total numbers of ratings on a given movie

Figure 3.2 is a histogram representing the total number of movies each user has rated. Figure 3.3 is a histogram representing total number ratings on a given movie. As there are many cases where a user has rated an only small fraction of movies or, movie is only rated by a very small fraction of users, the dataset is very sparse. In fact, the rating matrix is only 4.468% dense. To manage this issue, I have created a secondary dataset by filtering movies that are rated less than 100 times and users who have rated less than 100 times. The number of observations was decreased from 1,000,209 to 790,782, but the dataset got much denser as the secondary dataset was 13.551% dense. I will examine models in two different datasets to observe how performance varies.

movies.dat:

Genres attribute is a pipe-separated string where each movie had 1 to 5 genres. Every movie had at least one genre associated with it. Figure 3.4 shows the distribution of genres, “Drama” being the most frequent genre and “Film-Noir” being the least frequent genre.

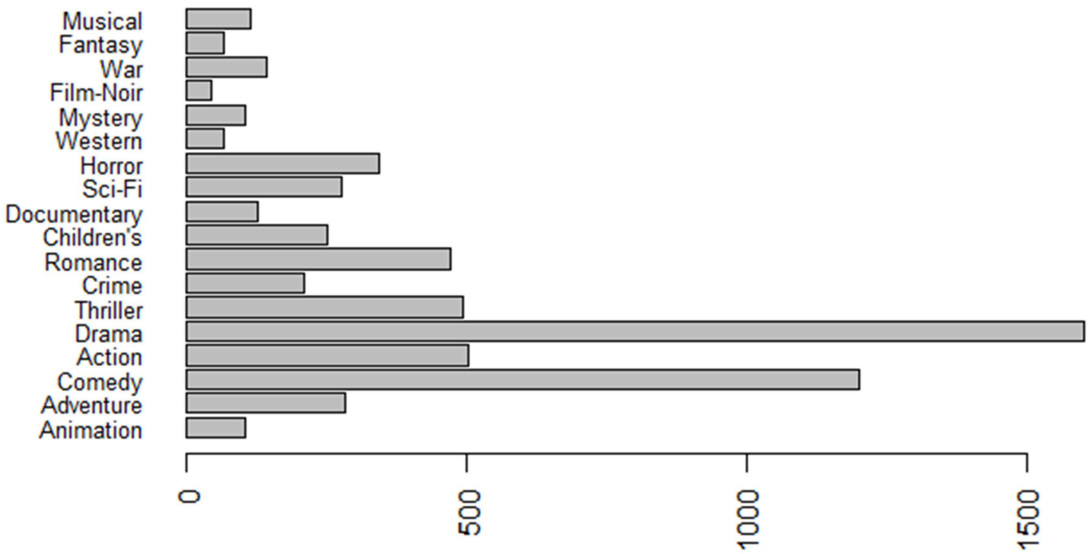

Figure 3.4 - Genre distribution of movies

4. Approach

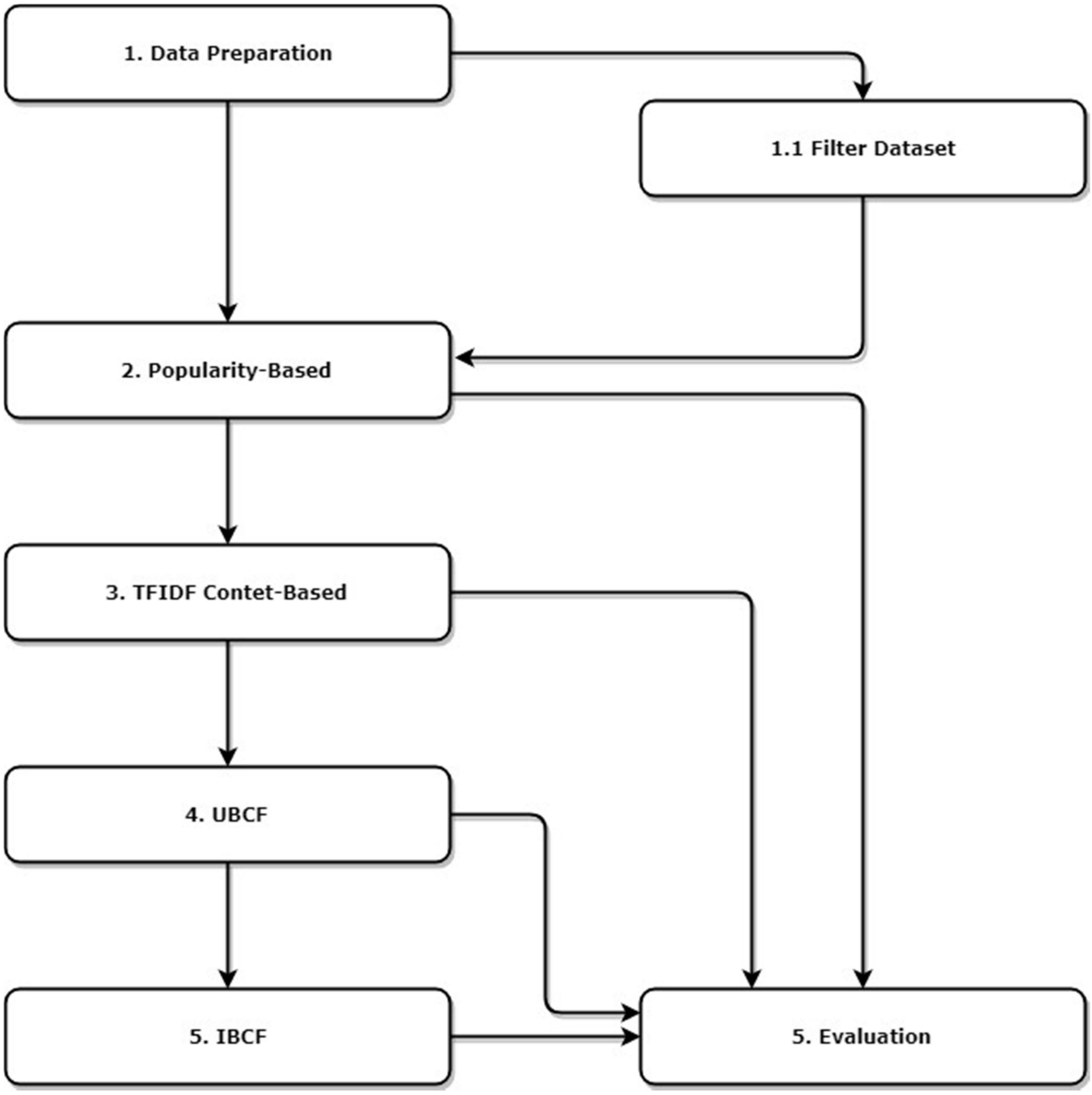

Figure 4.1 - Overview of the steps

Step 1: <Data Preparation>

Separate pipe-separated genres and create a genre matrix where each row represents a movie and each column represents a specific genre.

Split dataset randomly into 60% train, 40% test set. The model will try to predict ratings for test set and evaluation metrics will be computed by comparing test set rating data to the predicted ratings.

Create rating matrix where each row represents a user and each column represents a movie. The entry in the i-th row and j-th column will be 1 if user-i has positive rating on movie-j, -1 if negative, and NA if the user has not rated the movie.

Step 1.1: <Filter Dataset>

Prepare a secondary dataset by removing movies and users that has less than 100 ratings associated.

Otherwise, the dataset will be prepared following the same steps in step 1.

Step 2: <Popularity Based>

The popularity-based model is solely based on the popularity of a given movie. It will be used as a benchmark model. The model will try to predict a user's rating by examining if a given movie has a positive or negative rating on average.

Step 3: <TFIDF Content-Based Filtering>

The features that will be used for content-based filtering is movie genres. Since each genre has a different distribution, the relevance of each genre will be calculated by using TF-IDF. The relevance measure is then used to create a user profile representing each user's preference over certain genres, and the model will predict ratings by examining the genres associated with the given movie.

Step 4: <User Based Collaborative Filtering>

For UBCF, cosine similarity of each user is calculated using the rating matrix. Then the top k most similar user will be selected and used for making the prediction. The similarity measure will be used as weights, and different numbers of k will be tested to find the best performing number of k.

Step 5: <Item Based Collaborative Filtering>

For IBCF, cosine similarity of each item is calculated using the rating matrix. Then the top k most similar item will be selected and used for making the prediction. The similarity measure will be used as weights, and different numbers of k will be tested to find the best performing number of k.

5. Results and Discussion

5.1 Selecting optimal number of neighbors for IBCF and UBCF

Figure 5.1 shows accuracy and precision statistics for using different numbers of neighbors. For all of the four cases in both UBCF and IBCF, and both original and filtered datasets, accuracy and precision gains on adding more neighbors becomes negative or marginal after k = 30. For the rest of the analysis, k was chosen to be 30.

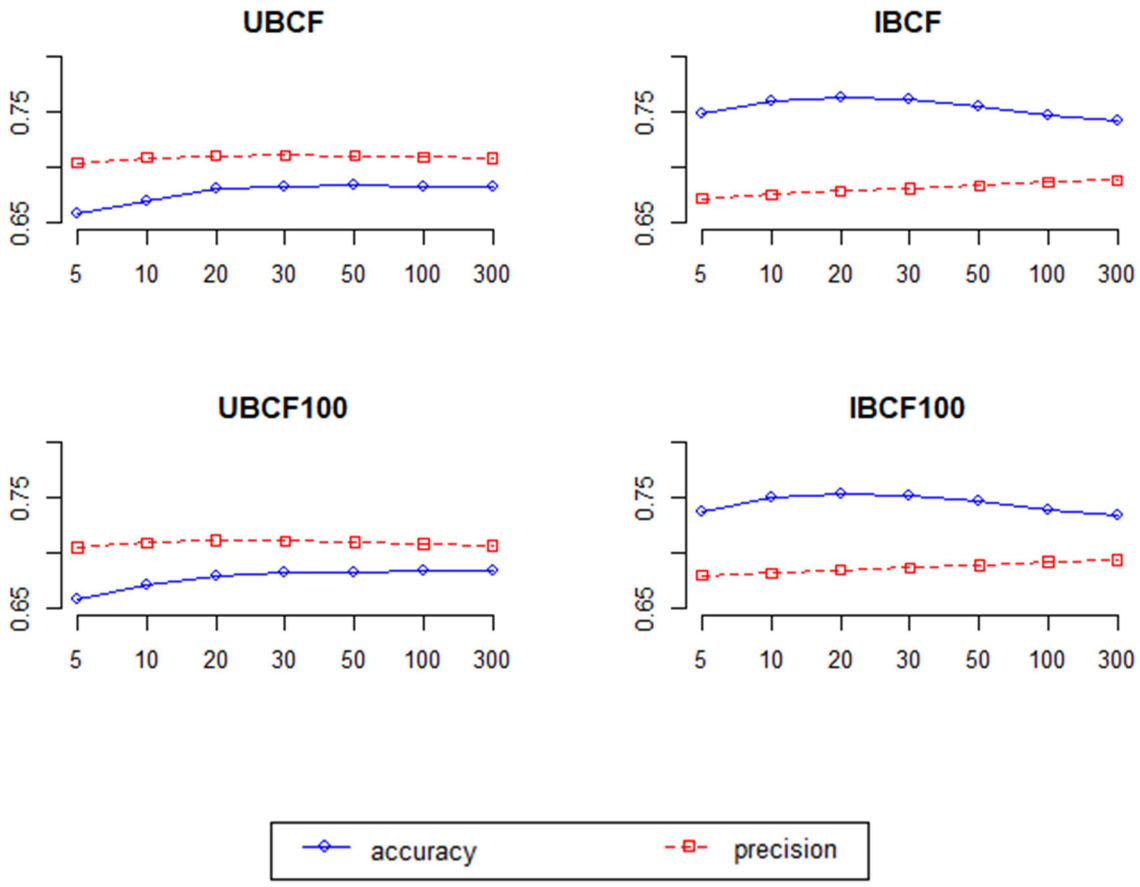

Figure 5.1 - Accuracy and Precision Statistics for Collaborative Filtering

5.2 Results

Original Dataset	Precision	Accuracy	Recall
Popularity-based	68.94%	65.39%	78.77%
TFIDF	67.78%	63.48%	75.12%
UBCF (k = 30)	**71.11%**	68.28%	84.44%
IBCF (k = 30)	68.10%	**76.08%**	**90.11%**

Table 5.2 - Evaluation metrics on the original dataset

Filtered Dataset	Precision	Accuracy	Recall
Popularity-based	68.78%	66.03%	79.44%
TFIDF	67.79%	63.75%	75.62%
UBCF (k = 30)	**71.28%**	67.84%	84.46%
IBCF (k = 30)	68.90%	**75.01%**	**90.26%**

Table 5.3 - Evaluation metrics on the filtered dataset

The primary evaluation criteria I chose were precision. Recommender systems aim to accurately predict given user's preferred item, and users do not observe the items that are not recommended to them. Hence the precision most accurately measures the response of the users to the system.

Accuracy and recall were used as secondary measures. The accuracy measures how accurate the model performed in general. The recall shows the proportion of good movies that are actually presented to the users, which relates to the amount of recommendation it can generate.

In overall, collaborative filtering methods performed relatively better, and TFIDF content-based method underperformed even compared to the popularity measure. It implies that movie genres as itself cannot make the most accurate prediction. Rather it is more useful when it is combined with other methods.

Precision-wise, User-based Collaborative Filtering method performed the best with it only being the model with precision over 70%. Directly comparing users to users would most accurately predict a given user's preference.

Item-based Collaborated Filtering showed the highest accuracy rate and the highest recall. IBCF was the most accurate model in general, but as the final model, UBCF would be more preferred as higher

precision is more related to user satisfaction. However, IBCF is still a very favorable model where it can generate a more exhaustive list of recommendations that would have been positively rated by a user.

5.3 Effect of Filtering the Dataset

Surprisingly, creating a secondary dataset and making rating matrix denser did not make an improvement over the original dataset. Possible reasons behind it could be:

1. The number of observations that were required to make an informed prediction was actually very small.
2. Although the number of users and movies that were lacking enough ratings were high, it only was a part of the whole rating matrix and takes up a relatively small amount of the whole.

6. Conclusion and Future Work

After examining different models for the movie recommendation, UBCF turned out to be the best recommender model in terms of precision. The recommendations made with UBCF are most likely to be accepted by the users. However, it is worth noting that IBCF could also be a good alternative or a good supplementary to UBCF depending on the purpose of the recommender system. For example, if one wants to instead to create a whole list of movie recommendations, UBCF would be a more favorable option because it can create more exhaustive lists of recommendations.

6.1 Possible Future Work

The topics I have covered in the report are only a small portion within the field of recommender system and there are many ways to expand the work. One possible way to expand could be to build models based on a larger movielens-20M dataset or IMDb dataset with broader scope of metadata available such as actors and tags information. In addition, more complex algorithms could also be implemented such as using classifiers, matrix factorization or even neural networks.

Reference

[1] P. Sappadla, Y. Sadhwani, & P. Arora (2017). Movie Recommender System: Project Report. [Online]. Available: https://pdfs.semanticscholar.org/767e/ed55d61e3aba4e1d0e175d61f65ec0dd6c08.pdf

[2] M. Zanitti, S. Kosta, & J. Sørensen (2018). A User-Centric Diversity by Design Recommender System for the Movie Application Domain. [Online]. Available: https://dl.acm.org/citation.cfm?id=3191580

[3] S. Yadav, Vikesh, Shreyam, & S. Nagpal (2018). An Improved Collaborative Filtering Based Recommender System using Bat Algorithm. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918308895

[4] R. Katarya, & O. P. Verma (2016). An effective collaborative movie recommender system with cuckoo search. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110866516300470