CS550 Massive Data Mining
Movie Recommendation system

Shreyas kulkarni
Rutgers, The State University of New Jersey
spk111@scarletmail.rutgers.edu

Udit Wadhera
Rutgers, The State University of New Jersey
uw17@scarletmail.rutgers.edu

ABSTRACT

Recommendation systems can now be found on a wide range of streaming websites. This project's purpose is to develop a movie recommendation system based on the 'MovieLens' dataset. We aim to merge the user's personalization with the movie's personalization, as well as other factors like genre, popularity, and so on. We accomplished this by employing a popularity model, a content-based model, a collaborative filtering model, and a latent components model. Hyperparameter tweaking, testing accuracy, and recommendation evaluation are all meticulously carried out for each model. By merging the predictions of latent factor based and collaborative filtering methods, we construct a combined linear model that improves the accuracy of projected ratings.To overcome the limitations of individual models, we combine all of the strategies to create a hybrid model that generates a final list of suggestions for each user. The hybrid model outperforms the distinct models in terms of quality and diversity of recommendations. Each model is thoroughly examined in the study.

KEYWORDS

SVD, CF, PBM, TMDB(The Movie Database), IMDB, User Rating

1 INTRODUCTION

As some people describe it, we are currently living in the "era of abundance." For any one product, there could be thousands of variations to choose from. Take a look at the examples above, which include streaming videos, social networking, and online shopping, to mention a few. Recommender systems aid in the personalization of a platform and the discovery of something the user enjoys.
The simplest and most obvious technique is to suggest the most popular products. However, to actually improve the user experience through personalized recommendations, specialized recommender systems are required.

A recommender system, like many other machine learning systems, generates predictions based on past behaviour of users. It entails predicting a user's choice for a collection of items based on previous experience. Content-based and Collaborative Filtering are the two most common ways for developing a recommender system.

Conference'17, Washington, DC, USA
© 2016 ACM. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Instead of relying on users' interactions and input, a **content-based approach** necessitates a large amount of information on the objects' inherent features. For example, movie properties such as genre, year, director, actor, and so on can be retrieved using Natural Language Processing, as can the textual content of articles.

Collaborative Filtering, on the other hand, requires nothing more than the users' previous preferences on a set of objects. Because it's based on historical data, the underlying premise is that consumers who have agreed in the past are more likely to agree again in the future. In terms of user preference, there are usually two options. An **explicit rating** is a rating given to an item on a sliding scale by a user, such as 5 stars for Titanic. This is the most direct way for users to express how much they enjoy a product. Indirect user preferences, such as page visits, clicks, purchase records, whether or not to listen to a music track, and so on, are reflected in **Implicit Rating**.

One of the most extensively used methods for Collaborative Filtering is Singular Value Decomposition, or SVD. SVD performs Matrix Factorization on the usermovie matrix to produce the relevant decomposition vectors demonstrating similarity between movies.

Facts of data:

- The grouplens movie review dataset was the main source of data for this research. MovieLens, a movie recommendation service, has a smaller dataset (ml-latestsmall) that describes 5-star rating and free-text tagging activity. There are 27,753,444 reviews from 283,228 users spread across 58,098 different movies in the dataset. Each rating is a number between 0 and 5. The timestamp of the review, the movie's genre, the keywords of user comments, and the IMDB and TMDB id for the related movie are all included in the dataset.

3 PROBLEM FORMALIZATION

In this project, we are developing a recommendation system using the MovieLens dataset. There are 3 tasks in this project namely Data selection and preprocessing, Rating prediction, and Item Recommendation.

Data preprocessing:

- Based on UserID, we separated the dataset into test and train categories, with 80 percent of each user's reviews used to train the model and the remaining 20 percent used to evaluate the model's accuracy. To do the aforementioned, the scikit-learn library's train test split function is utilized. The stratify attribute of the function specifies the feature that splits the dataset into testing and training.

- We combined the data set with the publicly available IMDB and TMDB datasets to obtain a better and detailed understanding of the movie for content-based recommendation and the importance of movie features, resulting in features such as release date, writer, director, and cast information, tag line, overview, popularity of the movie, and the rating and vote counts on their respective websites.
- Data cleaning comprised incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset and also the removal of redundant and non-contributing features from the dataset. Genres and other categorization traits are converted to a single vector.

Data Analysis:

(1) Average Number of rating compared to ratings given:

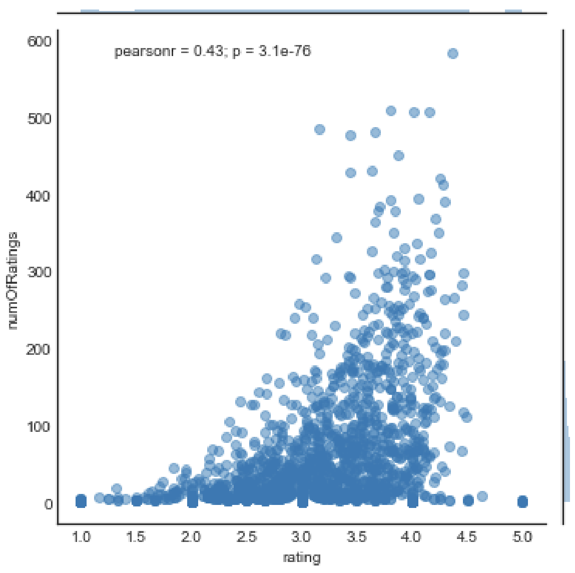


Figure 1: Number of rating compared to rating

(2) The Distribution of Movies by Genre

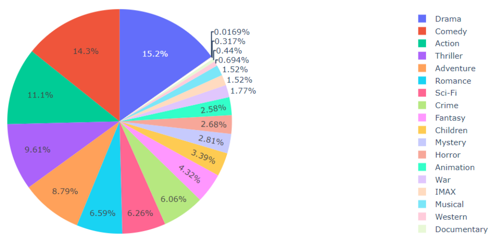


Figure 2: Movies by Genre

(3) Item frequency list:The distribution of item rating frequency is depicted in the graph below. In real-world circumstances, this distribution frequently satisfies a feature known as the long-tail property. Only a small percentage of the things are rated regularly, according to this feature. Popular things are those that are in high demand. The great bulk of things are only rated once in a while.
In most circumstances, high-frequency items are very competitive and offer minimal profit to the retailer. Lower frequency items, on the other hand, have higher profit margins. Many recommendation algorithms, on the other hand, prefer to propose popular things over infrequent ones. This behavior also has a detrimental impact on diversity, as consumers may become bored with the same set of popular item recommendations.
Matrix factorization technique to train model to understand user-item interaction by collecting user information in user latent factors and item information in item latent factors could be a solution for long-tail. Meanwhile, using the matrix factorization technique, we can minimize dimensionality and sparsity while also reducing memory footprint and making our system more scalable.

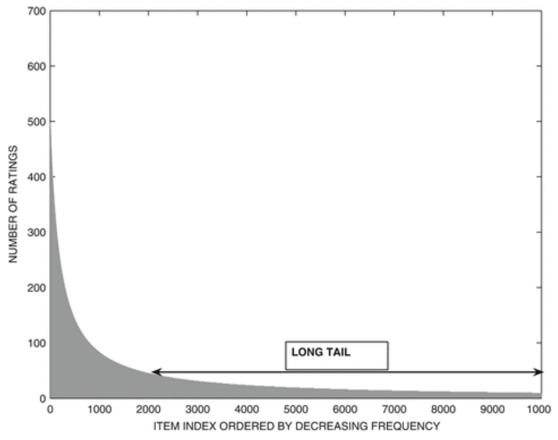


Figure 3: Item Frequency distribution

Rating Prediction: In this study, various models such as BaseLine, Slope One, SVD, KNN, Coclustering, and others were used. These models are attempting to predict the test dataset's items.

Item Recommendation: We forecast the rating for each movie and each user in the test data set in this step. Each user receives a top ten movie recommendation. In this task, we calculate precision, recall, F-score, and NDCG score.

4 THE PROPOSED MODEL

In this section, we will talk about models that we used for recommending the movies.

1. Popularity Model:

It's a form of recommendation system that works on the basis of popularity or anything that's currently popular. These algorithms look for products or movies that are currently trending or popular among consumers and then immediately recommend them.

- **Merits**: It doesn't have any cold start issues, which means it may recommend things on a variety of different filters from day one.
- **Demerits**:Not personalized, To every user, the system would recommend the same products/movies based only on their popularity.

We're attempting to provide recommendations based on popularity, which is determined by user actions such as amount of views, likes, release date, and watchlist additions, among other things. We're combining the MovieLens and TMDB datasets to create a new feature called popularity. We also take into account the weighted rating, which is calculated using the following formula:

$$W = R\frac{v}{v+m} + C\frac{m}{v+m}$$

where,
W = Weighted Rating
C = Mean vote average
m = minimum votes required
v = no. of votes for a movie
R = average rating of movie

2. Content based Recommendation:
A content-based recommender system seeks to guess a user's features or behavior based on the features of the item to which he or she responds positively.Once we know the user's preferences, we may use the feature vector generated to embed him/her in an embedding space and recommend him/her based on his/her preferences. The similarity metrics (which we'll get to in a minute) are calculated during recommendation using the item's feature vectors and the user's preferred feature vectors from prior records. The top few are then recommended.

In our model, we have tried to use features such as genres, release year and so on. We are using user vector and movie vector where user vector is given by average rating of user for particular genre and movie vector is given vector of zeros and ones with a one for relevant genre.

3. Collaborative Filtering:
Collaborative filtering employs similarities between individuals and objects at the same time to generate recommendations, which addresses some of the limitations of content-based filtering. This enables serendipitous recommendations, in which collaborative filtering models propose an item to user A based on the preferences of a user B who shares similar interests.

This method uses the ratings to form a neighbourhood N of a user and that neighbour is defined by the nearest neighbour approach with notion of similarity defined. We have used surprise library for prediction algorithms to analyze the performances of variants of KNN based models. KNNBasic takes the maximum number of neighbours into account and similarity metric as parameters. KNNwithMeans is very similiar to KNNBasic which takes mean ratings of each user into account. KNNwithZScore is also very similiar to KNNBasic which takes z-score normalization of each user. KNNBaseline takes basline rating of users.Similarity metrics are Cosine similarity, Mean squared difference, Pearson correlation and Pearson Baseline.

Variations of KNN based models:

- **KNNBasic:** A basic collaborative filtering algorithm is with the maximum number of neighbors to consider, k, and the similarity metric as parameters.
- **KNNwithMeans:** A basic collaborative filtering algorithm, similar to KNNBasic, that considers each user's mean ratings.
- **KNNWithZScore:** A simple collaborative filtering algorithm similar to KNNBasic that considers each user's z-score normalization.
- **KNNWithZScore:** A collaborative filtering algorithm that takes a baseline rating into consideration.

4. Latent Factor Method:
Latent Matrix Factorization is an algorithm that attempts to recommend the best items for each user given a set of m users and n items, as well as a set of user ratings for some items. This challenge comes in a variety of flavors and variations, the majority of which add extra dimensions to the problem, such as adding tags. What makes Latent Matrix Factorization so effective is that it produces extremely strong results from the core problem and can serve as an excellent starting point for further research.

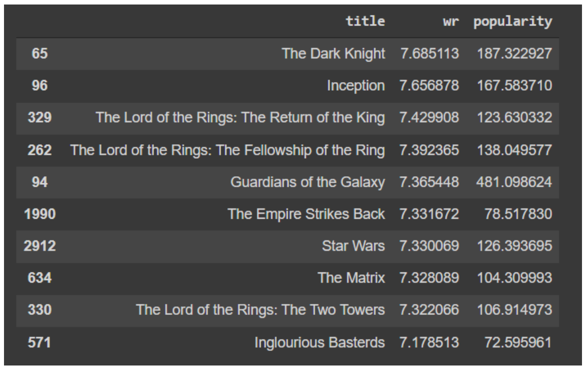
We want our system to predict unknown ratings as well and for this we are using latent factor method called SVD. SVD takes into account the lower dimensional representation of movies like people who like similar movies are mapped together. This model is used to form a problem to minimize the RMSE for unseen data. We have also used GridSearchCV to find the hyperparameters which gives us the lowest RMSE.

5. Combined Model:
From the earlier results, we saw that the algorithms like SVD, SVDpp, KNNBaseline and BaselineOnly are giving the best results and recommending the accurate movies for users. So, to improve our model further, we applied linear combination of the ratings from above methods. SVDpp introduces the implicit ratings while SVD balances the over-estimation of SVDpp. Baseline helps to introduce global biases of user and the items. KNNBaseline helps to introduce the item similarity notion for each user.

5 EXPERIMENTS

Firstly, in the popularity model, we are recommending movies according to popularity and weighted ratings. One of the example of recommendation is as follows:

In this image, we are showing popularity based recommendation of genre Action.



	title	wr	popularity
65	The Dark Knight	7.685113	187.322927
96	Inception	7.656878	167.583710
329	The Lord of the Rings: The Return of the King	7.429908	123.630332
262	The Lord of the Rings: The Fellowship of the Ring	7.392365	138.049577
94	Guardians of the Galaxy	7.365448	481.098624
1990	The Empire Strikes Back	7.331672	78.517830
2912	Star Wars	7.330069	126.393695
634	The Matrix	7.328089	104.309993
330	The Lord of the Rings: The Two Towers	7.322066	106.914973
571	Inglourious Basterds	7.178513	72.595961

Figure 4: Popularity based recommendation

We also tried comparing the time taken by each algorithm for implementation which is shown below:

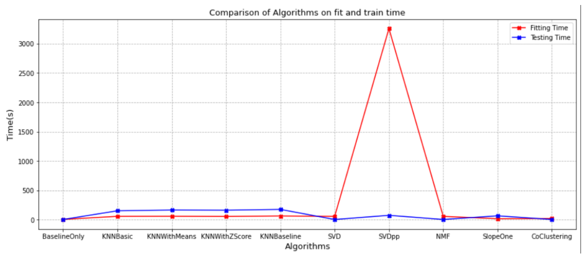


Figure 5: Comparison of Algorithms on fit and train time

As we can see in the image, SVDpp takes a lot of time for model fitting while all other algorithms take minimal time. Testing time is very similar for the algorithms.
We also did similarity metrics comparison which as follows:

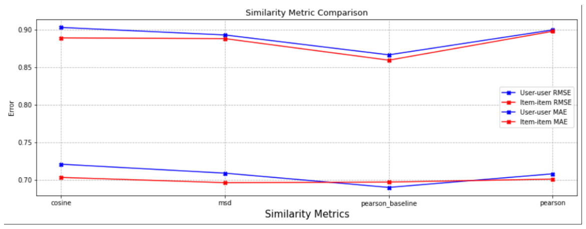
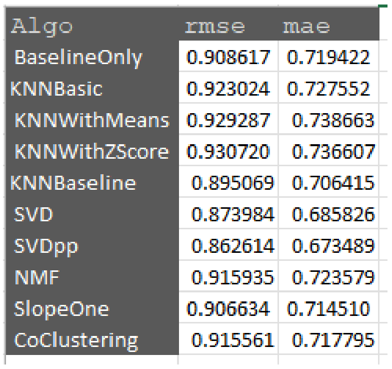


Figure 6: Similarity Metric Comparison

In the Rate prediction task, we are comparing the RMSE and MAE of all algorithms which is as follows:

SVDpp algorithms gives lowest RMSE and MAE score as compared to all other algorithms while KNNWithMeans gives the highest RMSE and MAE scores.



Algo	rmse	mae
BaselineOnly	0.908617	0.719422
KNNBasic	0.923024	0.727552
KNNWithMeans	0.929287	0.738663
KNNWithZScore	0.930720	0.736607
KNNBaseline	0.895069	0.706415
SVD	0.873984	0.685826
SVDpp	0.862614	0.673489
NMF	0.915935	0.723579
SlopeOne	0.906634	0.714510
CoClustering	0.915561	0.717795

Figure 7: Data points for comparison with diffrent alogorithms

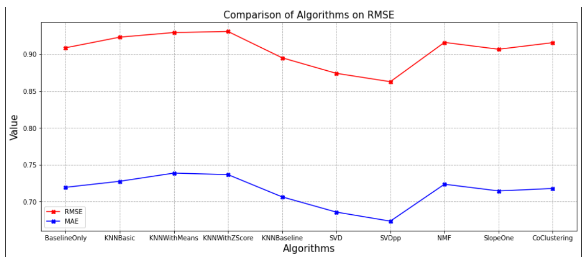
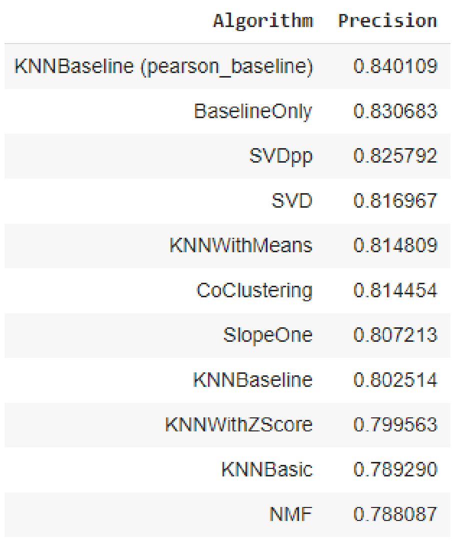


Figure 8: Comparison of Algorithms on RMSE and MAE

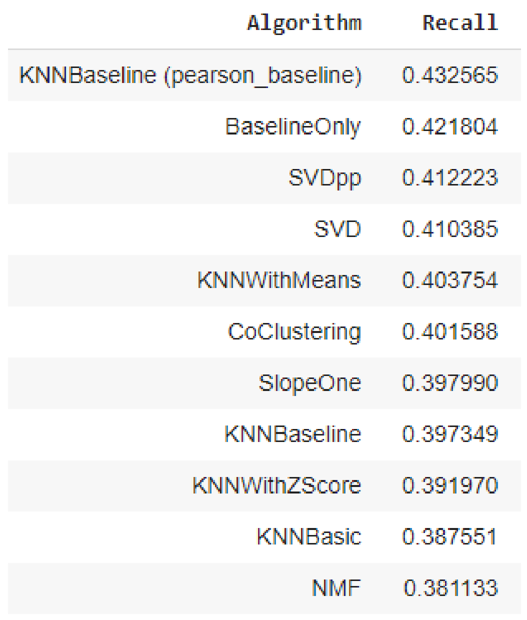
For the task 3, we did movie recommendations and evaluated using parameters as follows:



Algorithm	Precision
KNNBaseline (pearson_baseline)	0.840109
BaselineOnly	0.830683
SVDpp	0.825792
SVD	0.816967
KNNWithMeans	0.814809
CoClustering	0.814454
SlopeOne	0.807213
KNNBaseline	0.802514
KNNWithZScore	0.799563
KNNBasic	0.789290
NMF	0.788087

Figure 9: Comparison of Precision on Algorithms

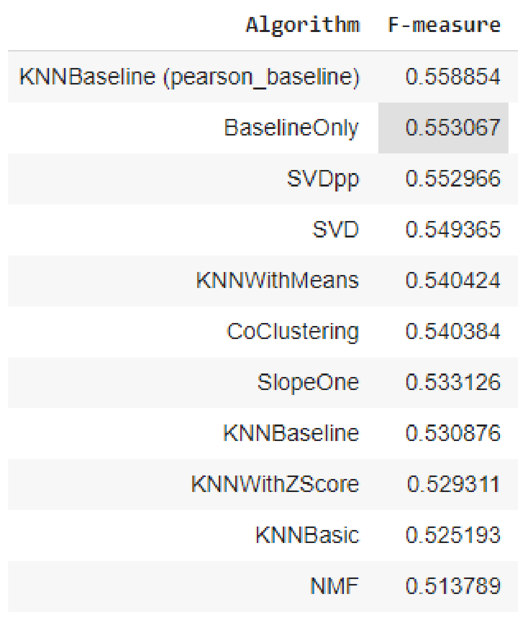
KNNBaseline gives the highest Precision amongst all algorithms.



Algorithm	Recall
KNNBaseline (pearson_baseline)	0.432565
BaselineOnly	0.421804
SVDpp	0.412223
SVD	0.410385
KNNWithMeans	0.403754
CoClustering	0.401588
SlopeOne	0.397990
KNNBaseline	0.397349
KNNWithZScore	0.391970
KNNBasic	0.387551
NMF	0.381133

Figure 10: Comparison of Recall on Algorithms

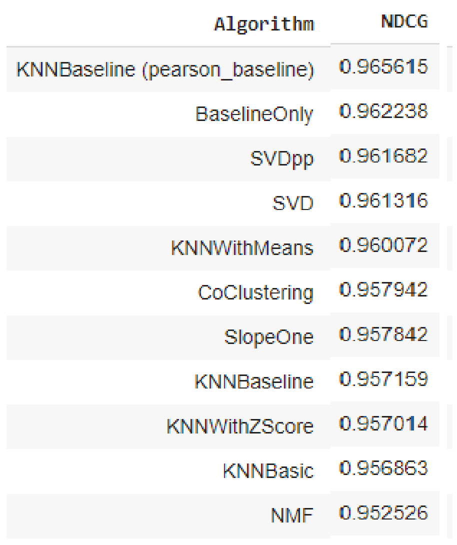
KNNBasic gives the highest value of Recall of all algorithms.



Algorithm	F-measure
KNNBaseline (pearson_baseline)	0.558854
BaselineOnly	0.553067
SVDpp	0.552966
SVD	0.549365
KNNWithMeans	0.540424
CoClustering	0.540384
SlopeOne	0.533126
KNNBaseline	0.530876
KNNWithZScore	0.529311
KNNBasic	0.525193
NMF	0.513789

Figure 11: Comparison of F-Measure on Algorithms

KNNBasic gives the highest value of F-measure of all algorithms.



Algorithm	NDCG
KNNBaseline (pearson_baseline)	0.965615
BaselineOnly	0.962238
SVDpp	0.961682
SVD	0.961316
KNNWithMeans	0.960072
CoClustering	0.957942
SlopeOne	0.957842
KNNBaseline	0.957159
KNNWithZScore	0.957014
KNNBasic	0.956863
NMF	0.952526

Figure 12: Comparison of NDCG on Algorithms

KNNBaseline gives the highest value of NDCG of all algorithms.

6 CONCLUSIONS AND FUTURE WORK

The content based genre is great when user has less ratings and they are decent for generating recommendation. Collaborative filtering and latent factor methods capture the user related features well as compared to other algorithms. As we combined the models of SVD and Collaborative Filtering, accuracy increases massively.

As we tried to combine the models which were giving best accuracy, we saw that test accuracy increased which gives us idea about how merging the models can increase the recommendation performance. We implemented models using small datasets. We can improve our recommendations further using large dataset.

ACKNOWLEDGEMENT

We would like to thank the Professor Mr. Yongfeng Zhang for his guidance as well as for providing necessary information regarding the project.

REFERENCES

1) Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In Proceedings of the 37th international ACM SIGIR conference on Research development in information retrieval (SIGIR '14). Association for Computing Machinery, New York, NY, USA, 83–92.

https://doi.org/10.1145/2600428.2609579

2) Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In Proceedings of the Web Conference 2021 (WWW '21). Association for Computing Machinery, New York, NY, USA, 1516–1527. https://doi.org/10.1145/3442381.3449973

3) Kumar, Manoj Yadav, Dharmendra Singh, Ankur Kr, Vijay. (2015). A Movie Recommender System: MOVREC. International Journal of Computer Applications. 124. 7-11. 10.5120/ijca2015904111.

4) C. M. Wu, D. Garg and U. Bhandary, "Movie Recommendation System Using Collaborative Filtering," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), 2018, pp. 11-15, doi: 10.1109/ICSESS.2018.8663822.