Comparison of Recommendation System on Movie Lens Data

Rushabh Bid[1], Fathima AlSaadeh[1], Keya Desai[1], and Naveen Narayanan M[1]
[1] Rutgers University
{rhb86,fya7,kd706,nm941}@scarletmail.rutgers.edu

ABSTRACT

In the era of big data, many companies are trying to take advantage of the enormous information available on their servers to detect and predict the users' behavior, and that led to developing various algorithms that can help with this process. Recommendation systems started to evolve in the mid-1990s to give the users better product recommendations based on their past browsing and rating history, and it is widely used by big companies like Youtube, Amazon, and Spotify. As this topic is becoming more and more important, we are experimenting in this project various famous models (e.g., kNN and SVD), focusing on the importance of hybrid models and building a novel Interest Sequence-Based CF model. [2, 11]

KEYWORDS

Recommendation Systems, Matrix Factorization, Data Analysis, kNN, SVD, Interest Sequence-Based CF Model

1 INTRODUCTION

Recommendation systems or engines are algorithms that are used to suggest relevant products to users. Nowadays, recommendation systems are used in a wide variety of applications, and they provide personalized online product or service recommendation for users such as movies, songs, restaurants, hotels, television programs, etc. Unlike the search engine, the recommendation engine tries to predict the rating or preference of the particular product for a given user. Over the last decade, more companies have started leveraging the use of these systems for improving the user experience in shopping and also to sell new products for relevant users. Recommendation systems are not only useful for customers but are also used by companies to learn about customer desires, current market trends, retain customers, create brand loyalty, and generate more revenue. Apart from these benefits, the data generated from the recommendation engines can be analyzed to take strategic and tactical decisions by the marketing/sales team of a company. The most popular two famous methods to approach recommendation systems are collaborative filtering and content-based recommendations. In Collaborative Filtering, for each user, recommender systems recommend items based on how similar users liked the item. In Content-based recommendation, recommendation systems work based on the detailed metadata about each item. We build an item profile for each item, and based on this item profile; recommendations are provided. In this project, the goal is to develop a various movie recommendation system for the movie review data set taken from MovieLens Latest Datasets. [4] This data set describes a 5-star rating from MovieLens, a movie recommendation service. We trained and tested several recommendation models such as baseline, SVD [7], kNN [1], Coclustering [5], SlopeOne [8], SVD++ [3] and Interest Sequence based Collaborative Filtering , using Python3 and other libraries as mentioned in the context.

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

2 DATA SELECTION AND PREPROCESSING

The first step in order to build a robust system was processing, analyzing and splitting the data into training, testing datasets to be used for training the models and evaluating it. As mentioned before we used python3 libraries: requests, zipfile, io, pandas, and sklearn.model_selection, to help us with this step.[9, 10] First, the system will check if the required files exist locally, otherwise, it will be downloaded from the original website [4], unzipped using requests, zipfile, io. Four main files will be used in this process shown in Figure (2). Ratings from different users on various movies are included in this data, Figure (1) shows the ER Diagram of these files.

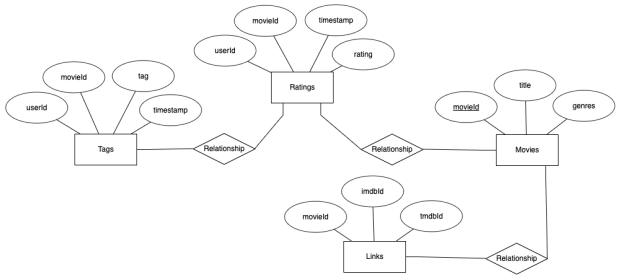

Figure 1: ER Diagram

There are about 102245 ratings, 9742 unique movies, 610 unique users, and 20 different genres. The "movies.csv" file is used to map the different movie ids with the movie title and its related genres. The "links.csv" file is to link the movieId with the corresponding Id of these movies in IMDB and TMBD. Figure (3) shows a snippet of the data set. The data was read in the correct format as a data frame. Then for each user, we randomly selected 80% of the user's ratings as the training ratings, and the remaining 20% ratings were used for testing our recommendation system. Before we start developing our recommendation algorithm, it is essential to do a preliminary analysis of our data set to have a statistical overview of the movie lens data set. As described above, the "ratings.csv" file consists of the rating details. We found that the mean and median rating of

```
├── ml-latest-small
│   ├── README.txt
│   ├── links.csv
│   ├── movies.csv
│   ├── ratings.csv
│   └── tags.csv
```

Figure 2: Data set Files

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

Figure 3: Data set snippet of ratings.csv

the data set is 3.501 and 3.5, respectively. Although the rating scale is from 0-5. The minimum and maximum ratings in our data set are 0.5 and 5 respectively. Figure (4) and Figure (5) shows the computed statistics and box plot of all the ratings given by different users on different movies. Apart from the basic analysis, we found few

```
print("Rating scale: 0-5")
print("Mean rating:", ratings['rating'].mean(axis=0))
print("Median rating:", ratings['rating'].median(axis=0))
print("Variance of rating:", ratings['rating'].var(axis=0))
print("Standard Deviation of rating:", ratings['rating'].std(axis=0))
print("Minimum rating:", ratings['rating'].min(axis=0))
print("Maximum rating:", ratings['rating'].max(axis=0))
```

```
Rating scale: 0-5
Mean rating: 3.501556983616962
Median rating: 3.5
Variance of rating: 1.08686721429614
Standard Deviation of rating: 1.0425292390605359
Minimum rating: 0.5
Maximum rating: 5.0
```

Figure 4: Basic statistics on ratings

interesting findings. Figure (6) shows the top 5 most viewed Movies by users in the data set along with their average ratings. This was calculated by counting the number of user id for each movie id and joining the result with the movies table. Our next analysis was based on the different genres. Figure (7) and Figure (8) shows the different plots that we made to analyses the genre distribution of the movies. From the above two graphs, it is evident that the genres comedy and drama had the highest number of movies, and only 6 out of the 20 genres had more than 1000 movies. The histogram of the rating distribution indicated by the blue bars tends to be normal. As expected, genres such as comedy, drama, and animation have a higher average rating compared to other genres, such as sci-fi,

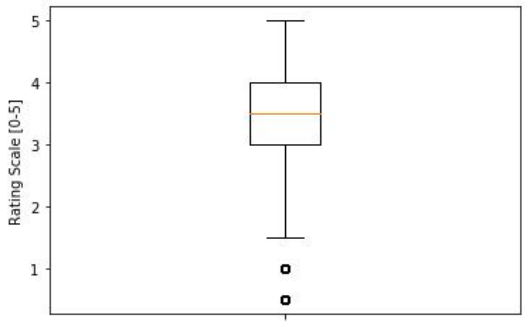

Figure 5: Rating Distribution - Box plot

movieId	title	genres	Average_Rating	Number_of_Ratings
356	Forrest Gump (1994)	Comedy|Drama|Romance|War	4.164134	329
318	Shawshank Redemption, The (1994)	Crime|Drama	4.429022	317
296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller	4.197068	307
593	Silence of the Lambs, The (1991)	Crime|Horror|Thriller	4.161290	279
2571	Matrix, The (1999)	Action|Sci-Fi|Thriller	4.192446	278

Figure 6: Top 5 Movies with most number of views

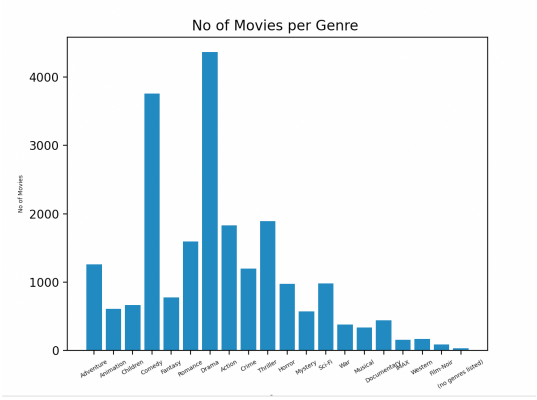

Figure 7: Number of movies in each genre

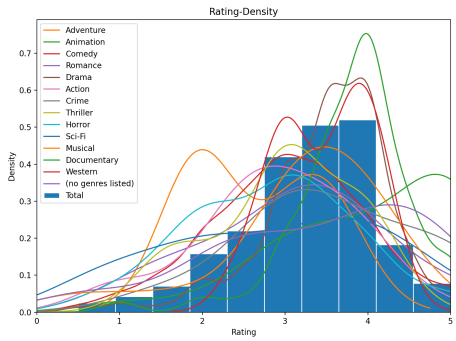

Figure 8: Genre-Rating Density

musical, and horror. The reason for this might be the fact that kids

and teens may not be interested in watching these genres, and they may not like it.

3 THE PROPOSED MODEL

In this project, we propose to develop a recommendation system for the movie's data set. The project involves three stages: Data pre-processing and analysis, Rating prediction, and Item recommendation. We have already discussed the stage of data pre-processing and analysis. Once the data set is divided as a training and testing data set, we can move to the next step of rating prediction. For this stage, we develop different recommendation algorithms using the training data set only. Once the model is developed, we can validate it using the test data. We use the model to predict the rating for each userId and MovieId combination in test data. We can evaluate our recommendation algorithm by estimating parameters such as MAE (Mean Average Error) and RMSE (Root Mean Square Error). MAE is an average of the absolute error between the predicted rating and the true rating. RMSE is the standard deviation of these errors. Hence MAE≤RMSE. We developed different recommendation algorithms and compared there MAE and RMSE values. The model with lower MAE and RMSE was preferred. Once the model is ready, we use this model to recommend movies to the user based on their past preferences. This step is known as Item recommendation. Item recommendation is made by sorting the list of movies for each user in the test data according to the predicted rating from the model. Then a list of top ten movies is generated for each user. It is crucial to make sure that our recommendation algorithm recommends new movies(movies in test data). Item recommendation of different models can be evaluated by calculating metrics such as Precision, Recall, and F-Score. These metrics are used to compare the performance of our model with true results. The movies in test data are ordered as per the true ratings and estimated ratings for each user. Precision is defined as the percentage of the testing items in the ten recommended items. The recall is the percentage of recommended testing items in all the testing items for the user. Precision and recall are calculated for each user, and then the average is computed for all the users. F-score is defined as the harmonic mean of precision and recall. F-score values have a range of 0 to 1. Higher the F-score, precision, and recall better is the recommendation system. The above process is shown in figure 9

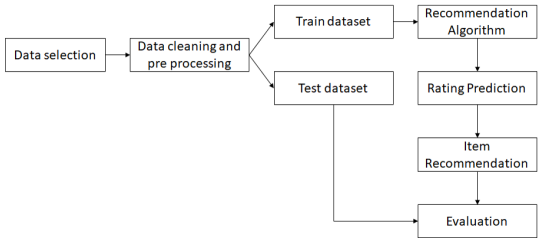

Figure 9: Flow Diagram

4 RATING PREDICTION

everal rating prediction models were used in this project in order to predict the items in the test dataset as these models don't know them, BaseLine, SVD, kNN, Slope One, Coclustering, interest sequence based collaborative filtering and hybrid models. Most of the above models were developed using surprise library.[6]

4.1 BaseLine Algorithm

This algorithm is designed to predict the baseline estimation for a given user and item. [6]

$$\hat{r_{ui}} = \mu + b_i + b_u$$

where $\hat{r_{ui}}$ is this predicted rating, μ is the average rating and b_u, b_i is the deviation for the user and the item. The training dataset is passed for fitting and training on the baseline model, and after that the testing dataset is passed to the model to return the predictions.

4.2 Singular Value Decomposition Algorithm

Singular Value Decomposition (SVD) algorithm, is used as collaborative filtering in recommendation system and it reduce the number of features of the data set by reducing space dimensions from m to n.

$$A = P\Sigma Q^T$$

Where P is the user-to-concept similarity matrix and Q is movie-to-concept similarity matrix

$$\hat{r_{ui}} = \mu + b_i + b_u + (q_i)^T p_u$$

where $\hat{r_{ui}}$ is this predicted rating , μ is the average rating and b_u, b_i is the deviation for the user and the item. The training data set is passed for fitting and training on the SVD model, and after that the testing data set is passed to the model to return the predictions.

4.3 k Nearest Neighbors Algorithm

k Nearest Neighbors(kNN) is an unsupervised and nonparametric machine learning algorithm to find clusters of similar items. Once the similar items are clustered, we can predict the rating of a movie as the average of the movies with similar ratings. For this project, we have considered two types of kNN algorithms: one that clusters similar users and the other one that clusters based on similar movies. kNN does not make any assumptions on the underlying data distribution but it relies on item feature similarity. When kNN makes an inference about a movie, kNN will calculate the distance between the target movie and every other movie in our data set. It ranks its distances and returns the top K nearest neighbor movies as the most similar movie recommendations. We can predict the rating as the average of the top k nearest neighbors. It is necessary to choose an optimal value of k and the correct similarity measure for clustering to get a good recommendation model. The kNN model in surprise package allows three different similarity measures: Mean Squared Difference(MSD), Cosine Similarity and Pearson-Baseline. Of the three similarities, the MSD similarity performed well and gave better results than the other two measures for the considered data set.

4.4 Slope One Algorithm

Slope one algorithm is one of the simplest collaborative filtering recommendation algorithms. It is based on a simple linear regression model. The slope one algorithm works on a rating data

sets (eg, rating scale of 1-10) and not on a binary data set (eg, Did you like it? "Yes" or "No"). The slope one algorithm was introduced by Daniel Lemire in 2005[8]. It uses a single free parameter, and its equation can be represented as

$$f(x) = x + b$$

Let U_i and V_j be the set of users who rated the movies i and j respectively. Then the rating of movie j by a User v is given by $R_{vj} = r_{ui} + d_{ji}$
where r_{ui} is the rating of item i by user u and

$$d_{ji} = \Sigma_{v \in U_i \cap U_j} \frac{R_{vj} - R_{vi}}{|U_i \cap U_j|}$$

In general, the accuracy of slope one algorithm is at par with other recommendation models.

4.5 Co-Clustering Algorithm

When user rates movies, there are two entities involved movies and users. Clustering in data mining refers to finding entities of similar type and grouping them together. The classical unsupervised clustering algorithm considers these two entities are separate and it does not take in to account the pairwise interaction between the two entities namely, users and movies. Our data can be represented as a matrix where the rows are the different users and column are the different movies. The value in each matrix is the rating given by the user for that movie. Co-clustering uses this matrix and tries to group similar users and similar movies based on the pairwise interactions.

4.6 Matrix Factorization Algorithm

The Matrix Factorization (MF) algorithm is another type of collaborative filtering algorithm and uses the idea of SVD algorithm. The user-item rating matrix is generated. This matrix is similar to the matrix used in Co-clustering and SVD algorithm. The basic idea was developed by Simon Funk in 2006[3]. The original Funk method was improvised, and various methods such as SVD++, Asymmetric SVD, Hybrid MF, and Deep Learning MF were developed. The general idea behind MF is to decomposed the user-item rating matrix into two lower dimensional matrices. The first matrix has one row for each user, and the second matrix has one item for each column. The algorithm can be expressed in mathematical terms as given below:

$$\tilde{R} = HW$$

where $\tilde{R} \in R^{\text{users} \times \text{movies}}$,
$H \in R^{\text{users} \times \text{latent factors}}$
$W \in R^{\text{latent factors} \times \text{movies}}$

According to Funk's Matrix Factorization, the predicted rating by user u for movie i is given by

$$r_{ui} = \Sigma_{f=0}^{nfactors} H_{uf} W_{fi}$$

Unlike the basic Funk's MF, the SVD++ algorithm takes into account the user and item bias and the predicted rating is given by

$$r_{ui} = \mu + b_i + b_u \Sigma_{f=0}^{nfactors} H_{uf} W_{fi}$$

where b_i and b_u are the bias for movie i and user u. We ran the SVD++-MF algorithm from surprise package on our data set.

4.7 Hybrid Approach

Previously in this section we looked at various recommendation methods that use ratings given to items by users and similarity between users to predict ratings for items that a user did not rate. Each of the approaches have their advantages and disadvantages and also they differ in performance. We considered the possibility to merge multiple approaches together to see if we could combine the predictions to improve the results. A hybrid approach was designed that would train multiple models on the same data and make predictions for the same test sets and then try to combine them in a way that captures the admirable qualities of all methods. One way of combining these predictions was to simply take the average of these multiple predictions. However, this would water down the best approach by combining it with the less accurate approaches. So, the way to do it was to weigh the approaches according to their significance levels. A mathematical description to combine ratings r_i using weights w_i is given by:

$$\hat{r} = w1 * r1 + w2 * r2 + w3 * r3...$$

such that,

$$\Sigma w_i = 1$$

The question to ask here is how do we decide the weights. We want the more accurate approaches to contribute more and the less accurate approaches to contribute less. We have evaluated the performance of each approach using error metrics. Let Err_i be an error value for the i^th approach. We cam choose the weights such that:

$$w_i \quad \alpha \quad \frac{1}{Err_i}$$

This seemed to be a promising approach after experimenting with different combinations of approaches and corresponding weights. The results are discussed in the next section 6.

4.8 Interest Sequence Based CF

This is a novel collaborative filtering approach modelling a more dynamic perspective over user behavior using the rating timestamps. With the correct impact identified, all recorded data can be factored in studying the user behavior. Most approaches that employ timestamps generally use them in a sense that decays the ratings over time. In simple words, ratings from the previous year are made more relevant than the ratings from the previous decade. A missing consideration is that the sequence in which the items were rated also describes a user's tendency to lean towards specific items. This sequence is formally known as the interest sequence of a user. In case of movie ratings, this plays a crucial role in describing the user behavior as it summarizes the evolution of user behavior and co-relation between movies (sequels, spin-offs,

etc.). The Interest Sequence Based Collaborative Filtering approach is inspired from 2016 Cheng et al[2].

Initially, we create the interest sequence for each user. The interest sequence is a sequence of ratings sorted by timestamps starting from the earliest. The length of all interest sequences can be different, but the algorithm does not rely on it. The interest sequence based similarity criteria for users looks for common sub-sequences between the interest sequences of the users. There are two main components of this similarity metric, namely Longest Common Sub Interest Sequence (LCSIS) and All Common Sub Interest Sequences (ACSIS). A Sub Interest Sequence is said to be common for the users, if the rated items were rated in the same order by time and the ratings did not differ by more than a predifined threshold value 'θ'. $|LCSIS|$ is defined as the length of the longest common sub-sequence between the users. $|ACSIS|$ is defined as the number of common sub-sequences (of any length) between the users.

Since, the number of movies rated by users can vary a lot and consequently we see variability in the interest sequences and sub-interest sequences. The values $|LCSIS|$ and $|ACSIS|$ are normalized to counter-balance this variability:

$$SIM_{LCSIS}(u,v) = \frac{|LCSIS(u,v)|}{\sqrt{|LCSIS(u,u)| * |LCSIS(v,v)|}}$$

$$SIM_{ACSIS}(u,v) = \frac{|ACSIS(u,v)|}{\sqrt{|ACSIS(u,u)| * |ACSIS(v,v)|}}$$

Once, the two similarity metrics are normalized, their importance is factored in the similarity between users by 'α'. The value of α is depends on the application. The similarity is given by:

$$SIM(u,v) = \alpha * e^{SIM_{LCSIS}(u,v)} + (1-\alpha) * e^{SIM_{ACSIS}(u,v)}$$

Once the similarity between users is obtained, the rating prediction is similar to the generic CF approaches. 'K' users are selected that rated the subject item and are most similar to the subject user. A weighted average is calculated for the deviation of the subject item rating from the mean rating for these 'K' users. This value is added to the mean rating of the subject user to predict the subject item rating. However, this approach is computationally expensive when similarity between users is calculated. This is a result of the novel interest sequence structure and the requirement to find sub-sequences between them. Such requirements don't allow the computation via quick matrix operations and thus requires heavy computational resources. The data-set was shrunk to 25% size for this approach.

5 ITEM RECOMMENDATION

The next step after rating prediction is to recommend items for each user based on their past preferences. In this step, we predict the rating for each movie and each user in the test data set. Once the rating is predicted, we arrange the movies according to the predicted rating and filter for the top 10 movies. Similarly, we filter for the top 10 movies from the test data set for each user. Now we have to compare the two sets of recommended movies. We calculate various parameters such as precision, recall, and F-score for each user, as discussed before. Once we have calculated the following metrics for each user, we take averages for these parameters across all users.

6 RESULTS AND ANALYSIS

In this section, we document the results obtained for all the models discussed in Section 4. The metrics of RMSE and MAE are used to evaluate the accuracy of rating prediction and metrics such as Precision, Recall and F-Score are used to evaluate the accuracy in Item recommendation to each user. Computation of RMSE and MAE is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_{ui} - \hat{r_{ui}})^2} \quad (1)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|r_{ui} - \hat{r_{ui}}| \quad (2)$$

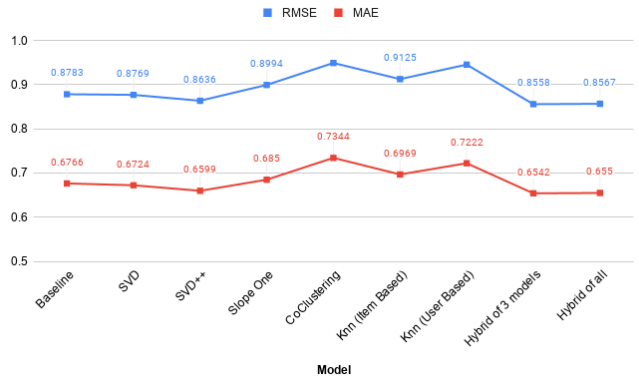

Figure 10: RMSE and MAE values of each model

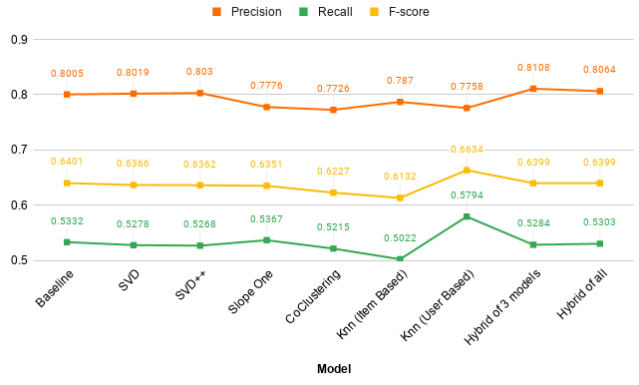

Figure 11: Precision, Recall and F-score of each model.

The results of the experiments are given in table 1. Out of all the models, SVD++ gives the least error, (RMSE 0.8636) followed by

Model	RMSE	MAE	Precision	Recall	F-score
Baseline	0.8783	0.6766	0.8005	0.5332	0.6401
SVD	0.8769	0.6724	0.8019	0.5278	0.6366
SVD++	0.8636	0.6599	0.803	0.5268	0.6362
Slope One	0.8994	0.685	0.7776	0.5367	0.6351
CoClustering	0.949	0.7344	0.7726	0.5215	0.6227
kNN (Item Based)	0.9125	0.6969	0.787	0.5022	0.6132
kNN (User Based)	0.9453	0.7222	0.7758	**0.5794**	**0.6634**
Hybrid models					
SVD + SVD++ + Baseline (weights: 0.3, 0.6, 0.1)	0.8588	0.6570	0.8090	0.5294	0.6400
SVD + SVD++ + kNN (Item based) (weights: 0.3, 0.6, 0.1)	0.8577	0.6554	0.8085	0.5294	0.6399
SVD + SVD++ + SlopeOne (weights: (1/rmse + 1/mae) ^12)	**0.8558**	**0.6542**	**0.8108**	0.5284	0.6399
Hybrid of all 6 models					
weights: 1/rmse	0.8609	0.6596	0.8062	0.5263	0.6369
weights: 1/mae	0.8608	0.6595	0.8057	0.5262	0.6366
weights: (1/rmse + 1/mae) ^10	0.8567	0.6550	0.8064	0.5303	0.6399

Table 1: Experiments Results

SVD model (RMSE 0.8769). The baseline model (RMSE: 0.8783) has a better performance than every model except SVD and SVD++. For the kNN algorithm, two methods are considered: user-based kNN and item-based kNN. In user-based kNN, similarities is computed between users and in item-based kNN, it is computed between items. For the movie lens dataset, item-based kNN (RMSE: 0.9125) performs much better than user-based kNN (RMSE: 0.9453). Intuitively, we hope for the Item-Item CF to perform better because in a realistic scenario there are more users using an item as compared to items used by a single user. More data allows us to formulate data-driven similarities between items and hence a better behavior summary and consequently better performance is observed. These trends can be observed from Figure 10. Further experiments are conducted by building hybrid models using combinations of six algorithms as described in Section 4. For predictions using this method, weighted average of ratings for each (user, item) pair from different models is taken. The weights are tweaked to give importance to particular models more than the others based on the error values. Initially, the weights were set inversely proportional to RMSE or MAE. The low range of variation in the error values for the different approaches did not bring the justified weight for each approach. The significance needed to be maximized. The inverse of both the error metrics were added, but this was not significant enough. Later, the range of significance was magnified by taking inverse proportions of powers of these error metrics and positive results were observed.

On trying different combinations of algorithms and tweaking the weights of each algorithm, it is observed that combining SVD with SVD++ improves performance. The results of some combinations are given in table 1. The least RMSE of 0.8558 is achieved by a hybrid model that combines SVD, SVD++ and SlopeOne, with most weight assigned to SVD++ of 0.47, followed by SVD of 0.33 and least to SlopeOne of 0.19. These weights are obtained by:

$$w_i = (\frac{1}{RMSE_i} + \frac{1}{MAE_i})^{12}$$

From the results given in table 1, we observe that the range of precision, recall and F-score are almost the same for all the models. As in most data science problems, we have a trade off between precision and recall. When we try to improve recall, the precision decreases and when we improve precision, recall decreases. So going for a higher precision or recall is always determined by the application or use case. So in cases where we need optimal values, we can combine the two metrics to get F-score. F-score increases if the two values are almost same. In order to get a balanced model, we try to maximize the F-score. In our case, F-score is maximum for kNN (User Based) model. The trends in precision, recall and F-score for the different models can be observed in figure 11.

K	alpha	RMSE	MAE	Precision	Recall	F-score
1	0.4	1.1307	0.8748	0.7226	0.4819	0.5782
1	**0.8**	**1.1271**	**0.8714**	0.7242	0.4841	0.5803
2	0.8	3.6758	0.9181	0.7274	0.4790	0.5776
10	0.8	9.8735	1.2300	**0.7321**	**0.4878**	**0.5855**
50	0.8	10.8957	1.4465	0.7308	0.4710	0.5728

Table 2: Results for Interest Sequence based CF

Table 2 depicts the performance of the Interest Sequence Based CF approach. Experimentally, it was observed that higher values of α worked better for our application. This suggests that the length of the longest common sub-sequence summarizes the user behavior more significantly as compared to the number of common sub-sequences. We also observe that the performance decreases with the increase in 'K'. This is a result of the decreased data-set size. As mentioned in Section 4, the data was shrunk to 25% size. The

number of users decreases and consequently so does the probability of finding similar users that rated the same user. The prediction for some User-Item pairs can be very realistic, but when we extend it to evaluate the performance for the entire data, the performance reduces. Limiting the data leads to less informed predictions as the user behavior cannot be summarized efficiently. We observe from Table 2 and Figure 12, that the lowest error rate for 'K' = 1 and 'α' $\geq$ 0.8. In the availability of sufficient data and computing resources, this approach can be promising with scope for further experimentation with the parameters involved. Once again in table 2 it is observed that the range of precision, recall and F-score are almost the same for different values of 'α' and 'K'. The highest F-score is obtained when 'K' = 10 and 'α' = 0.8. For this value of K and 'α', the MAE and RMSE values are high. In this case it is better to give more importance to rating prediction errors because, we have considered only a subset of the data set (25%) and as a result we will get almost similar precision and recall values for the different values of 'K' and 'α'.

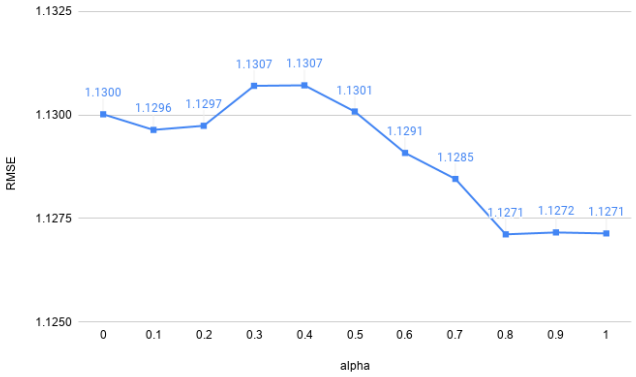

Figure 12: RMSE vs α for interest sequence model (K = 1)

7 CONCLUSION AND FUTURE WORK

In this project, we used for experimenting and testing a 5-star rating from MovieLens with 102245 ratings, 9742 unique movies, 610 unique users, and 20 different genres; we built the models and methods in a way that make it work with other data sets and can scale for more massive data sets. Experimenting on various models before building the recommendation system was essential for us to learn more about the models and on the other side to have scientific numbers (RMSE and MAE) that convince us which model for recommending the items to the users. Predicting ratings using these different models: Baseline, SVD, SVD++, Slope One, Co Clustering, kNN, Hybrid models, gave us a noticeable conclusion that the hybrid model (SVD, SVD++, and Slope One) have the lowest error. Combining three predicting models made us benefit from their complementary advantages; as a result, it achieved the highest precision for the items recommended. We tried to take this project to a novel level by building the Interest Sequence-Based collaborative filtering model from scratch. Refining this model to make it faster and work on the larger counterparts of the data set is one of our main future improvements for this recommendation system as we see promising results on the scarce data and computing resources at our disposal. The Interest Sequence Based CF has an advantage over the other methods discussed and we hope to consider this as an addition to the hybrid approach pushing its boundaries even further.

ACKNOWLEDGEMENT

We cannot express enough thanks to Rutgers University faculty staff for the resources and knowledge, professor Yongfeng Zhang and teaching assistant Shuyuan Xu. We offer our sincere appreciation for the learning opportunities provided by the faculty staff. Our completion of this project could not have been accomplished without the team collaborative hard work along with the support of all the referenced resources.

REFERENCES

[1] N.S. Altman. 1991. An Introduction to Kernel and Nearest Neighbor Non parametric Regression. *Cornell University* (1991).
[2] Dong Y Dong H Zhang W Cheng W, Yin G. 2016. Collaborative Filtering Recommendation on Users' Interest Sequences. *PLoS ONE 11(5): e0155739.* (2016).
[3] Simon Funk. 2006. Matrix Factorization. (2006). https://sifter.org/~simon/journal/20061211.html
[4] GroupLens. 2020. MovieLens Latest Datasets. (2020). https://grouplens.org/datasets/movielens/latest/
[5] Mohamed Nadif Gérard Govaert. 2013. *Co-Clustering: Models, Algorithms and Applications.* Wiley ISTE.
[6] Nicolas Hug. 2016. Surprise. (2016). https://surprise.readthedocs.io/en/stable/
[7] Jim Lambers. 2010. The SVD Algorithm. (2010). https://web.stanford.edu/class/cme335/lecture6.pdf
[8] Maclachlan A. Lemire, D. 2009. Slope One Predictors for Online Rating-Based Collaborative Filtering. *SDM* 8 (2009), 1–5.
[9] Python. 2009. Pandas. (2009). https://pypi.org/project/pandas/
[10] scikit-learn developers. 2007. Scikit-Learn. (2007). https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[11] Richa Sharma and Rahul Singh. 2016. Evolution of Recommender Systems from Ancient Times to Modern Era. *Indian Journal of Science and Technology* (2016).