CS 550 Final Project Recommender system

Abhilash Kolluri
Rutgers University
ak2048@scarletmail.rutgers.edu

Pruthviraj Patil
Rutgers University
pp865@scarletmail.rutgers.edu

Manad Desai
Rutgers University
md1713@scarletmail.rutgers.edu

ABSTRACT

Recommendation Systems have become the disguised backbone of many companies. They have become the break-or-make factor for emerging companies across the globe. It is because, the better the Recommendation Systems, the better the suggestion/recommendations, which helps attract new customers and retain old subscribers on platforms like Netflix, Spotify, etc. Given the vast database of products or movies these companies offer, it is challenging to provide state-of-the-art recommendations. Our Project aims to build a Recommendation System that provides suggestions based on three algorithms: User-based collaborative filtering, Model-based collaborative filtering (SVD), and neural collaborative filtering (NCF). Our Project uses the Movie lens dataset and makes use of metrics like Precision, Recall, f1, and ndcg to compare the implemented algorithms.

KEYWORDS

Recommendation Systems, User-based collaborative filtering, Model-based collaborative filtering, SVD, neural collaborative filtering, Movie lens dataset, Precision, Recall, f1, ndcg.

1 INTRODUCTION

The usage of Recommendation Systems is growing exponentially in today's world. The need for a better and more efficient Recommendation System is the current need of the hour and there are various ongoing research projects in the same. It should be fairly simple to convince someone of the importance of this. Looking at various streaming platforms like Amazon Prime, Netflix, Disney Hotstar, etc should give some idea of that. Recommendation Systems have penetrated the e-commerce space and other various social platforms like Instagram and Twitter. Not to forget, the Netflix challenge, to come up with the best collaborative filtering algorithm to predict the user rating ratings for films that had a cash prize of US$1,000,000 talks volumes about the importance of Recommendation Systems. Data is the new Oil. The above statement holds and makes immense sense for Recommendation Systems indeed. With the advent of data, recommendation systems have become an exciting field, and techniques like deep learning started to play their part in Recommendation Systems[1].

In this project, we have used the movie lens dataset. The dataset is partitioned into train and test with 80% being the train data and 20% being the test data. We have built three recommendation systems with the movie lens dataset. The performance of these models was judged based on the metrics of precision, recall, f1, and ndcg. After computing these values, we have found that Neural Collaborative Filtering(NCF) performed the best, followed by SVD and lastly User to User Collaborative filtering. We have used Cosine Similarity and Pearson Correlation techniques for User to User Collaborative Filtering. The exact values of the above models will be described in the Result section of this report.

The further of Project Report is divided into the following sections: 2) Dataset Used, 3) Related Work, 4) Proposed Models, 5) Rating Predictions (RMSE and MAE calculations for various models and comparison), 6) Results and 7) Conclusion and future work.

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

2 DATASET

. For this project, we worked on moviesLense latest small dataset. This dataset consists of 100,000 ratings on 1700 movies by 1000 users. It is a sparse dataset as not every user has seen all the movies. After doing dataset explorations, we come to know that there are 19 different genres of movies in this dataset such as 'Musical', 'Fantasy', 'Documentary', 'War', 'Horror', 'Animation', 'Sci-Fi',etc. After diving into the dataset, we came across some of the very interesting things about the dataset. As you can see from the graph[1] highest number of ratings is 4. In addition to this, we can see in Figure[6] the highest number of times watched movies.

. For this experiment, we have removed unnecessary columns from the dataset to reduce the computation. For better prediction, we only considered movies with at least 50 ratings. We also normalized the rating for each user so we can make a better rating prediction. We mainly focused on attributed such as UserID, MovieID, and Rating for this experiment. For training purposes, we have used 80% of the data and for the testing purpose, we have utilized the rest of the data.

3 ALGORITHMS

In this section, we will briefly discuss all the algorithms, we have used in our project.

3.1 User to User Collaborative Filtering

We have implemented User to User collaborative filtering which uses user similarity to recommend items. Users similarity is calculated using their behaviour for our example for movie lens dataset, users who have similar taste of movies are similar. The User to User collaborative Filtering is really useful when enough knowledge or domain expertise of the items are not available. The model can also help us discover new interest or popular items among group of user. However the User to User Collaborative Filtering suffers with the problem of cold start. There are multiple ways with which we can calculate user similarity for this project we have

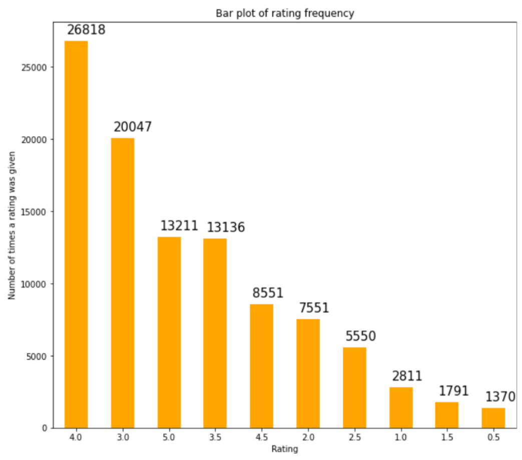
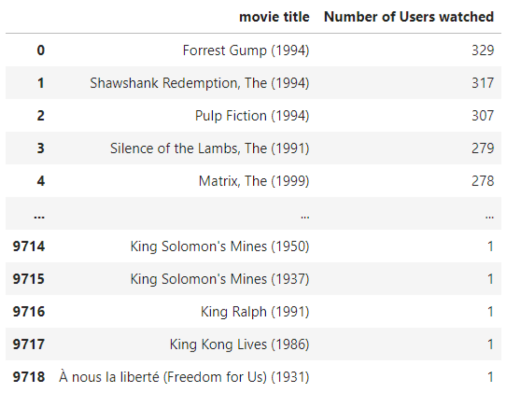


Figure 1: Number of ratings given by user



	movie title	Number of Users watched
0	Forrest Gump (1994)	329
1	Shawshank Redemption, The (1994)	317
2	Pulp Fiction (1994)	307
3	Silence of the Lambs, The (1991)	279
4	Matrix, The (1999)	278
...	...	...
9714	King Solomon's Mines (1950)	1
9715	King Solomon's Mines (1937)	1
9716	King Ralph (1991)	1
9717	King Kong Lives (1986)	1
9718	À nous la liberté (Freedom for Us) (1931)	1

Figure 2: Highest rated movies

calculated similarity using Cosine similarity and Pearson correlation.There are multiple steps we followed to implement user to user recommendation system, below are the steps.

3.1.1 Data preprocessing. We divided the dataset into test and training sub datasets with a spit ratio of 0.2. We removed the attributes which are not used for processing. Movies who had less than 50 user ratings were filtered out. We normalized the ratings of each user by extracting average rating of each user, this change was done because many users have a tendancy to rate high values.Create a user- item matrix dataframe with rows as userIDs and columns as movieIds.

3.1.2 ***Cosine Similarity Calculation.*** Cosine similarity between two users can be calculated by considering their row vectors in the user item matrix and calculating the cosine angle between the two n dimensional vectors. The angle can be calculated with below formula

$$S_c(A, B) = cos(\theta) = \frac{A.B}{||A||||B||} \tag{1}$$

The distance between vector of user A and user B gives us a good estimate of how similar the users are. Cosine angle between them

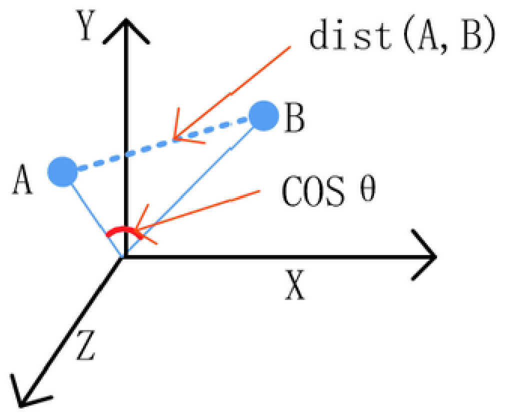


Figure 3: Cosine Similarity

helps us to measure the distance between them. If the vectors are coinciding means that the angle between them is 0 indicating that the users are similar. An angle of 90 shows they are unrelated and an angle of 180 show they have opposite tastes.The value of cosine similarity ranges from -1 to 1.

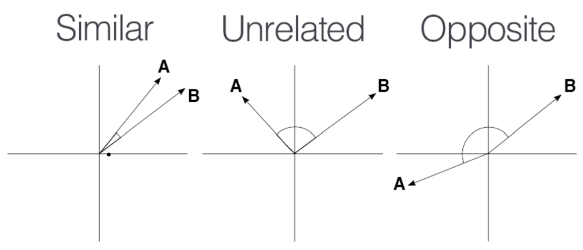


Figure 4: Cosine Similarity angle difference

3.1.3 ***Pearson correlation.***

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}} \tag{2}$$

Similar to cosine similarity the pearson correaltion has a range from -1 to 1. If the correlation is 1 the users are perfectly similar , if its -1 then they have opposite tastes.

3.1.4 ***Weighted average/K nearest similarities.*** Even after cleaning the data and calculating similarities data may still contain outliers which would affect the rating prediction. Taking similarity of all users is not efficient. So we only consider similarity of top K users. The value of K needs to be chosen wisely for our project we have chosen value of 100.

3.1.5 ***Prediction.*** Once we get the similarities for each user we can predict the rating for the missing values for each user.

$$y_{A,M} = \frac{S_{A,B} * R_{B,M} + S_{A,C} * R_{C,M} + \ldots S_{A,K} * R_{K,M}}{S_{A,B} + S_{A,C} + \ldots S_{A,K}} \tag{3}$$

$\gamma_{A,M}$ = Average predicted Rating of user A for movie M
$S_{A,B}$ = Similarity of user A with user B
$R_{B,M}$ = Rating of user B for movie M
$S_{A,K}$ = Similarity of user A with K

From the results we found out that cosine similarity outperformed Pearson Correlation for our data. The MAE and RMSE for cosine similarity were 1.98 and 2.12 while RMSE for Pearson Correlation were 2.05 aand 2.27. We have noted down the results for other metrics in the below Results section.

3.2 Singular Value Decomposition(SVD)

For this project, we have used the Singular Value Decomposition technique(SVD)[3] which is a Collaborative Filtering Approach, as our 2nd approach to implementing Recommendation System. In short, SVD[2] is a matrix factorization technique. The given matrix is broken down into a product of three matrices. U, Σ and V^T. Here Σ is the diagonal matrix. That means, all the values are present along the diagonal only. Typically the input matrix consists of rows that represent the users and columns that represent the rating the user has provided. The user matrix is decomposed into 3 matrices. The first matrix U represents the user, the last matrix V represents the items and the diagonal matrix conveys the relationship between them.

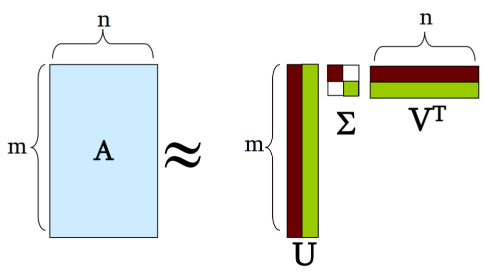
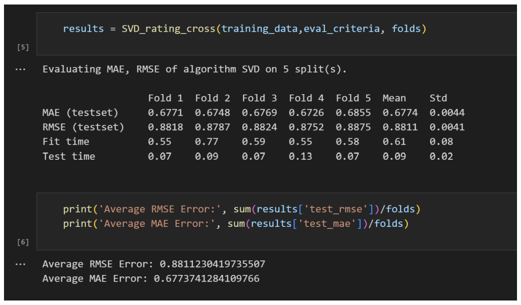


Figure 5: SVD

Now that we understand the basics of SVD, there are certain advantages SVD offers. Firstly, since it reduces the dimension, this approach is beneficial for a large data set. Secondly, it reduces the memory footprint and computation cost. Finally, it reduces the noise that is present in the input data and de-mystifies the recommendation approach. Coming to the implementation part, the SVD makes sense to use on the Movielens dataset. The dataset has a dimension of 600x9000. Analyzing the data by reducing the dimension provides a deeper understanding of how users and ratings are related.



```
results = SVD_rating_cross(training_data,eval_criteria, folds)
[5]
...  Evaluating MAE, RMSE of algorithm SVD on 5 split(s).

                     Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
     MAE (testset)   0.6771  0.6748  0.6769  0.6726  0.6855  0.6774  0.0044
     RMSE (testset)  0.8818  0.8787  0.8824  0.8752  0.8875  0.8811  0.0041
     Fit time        0.55    0.77    0.59    0.55    0.58    0.61    0.08
     Test time       0.07    0.09    0.07    0.13    0.07    0.09    0.02

print('Average RMSE Error:', sum(results['test_rmse'])/folds)
print('Average MAE Error:', sum(results['test_mae'])/folds)
[6]
...  Average RMSE Error: 0.8811230419735507
     Average MAE Error: 0.6773741284109766
```

Figure 6: SVD Results

Fig. 6 shows the code snippet showing the prediction and recommendation using SVD. We have achieved a Precision of 19.45%, a Recall of 11.20%, and an F1 score of 14.21%. The RMSE and MAE values were 0.77 and 0.65 respectively. This is very low compared to the previous implementation using Cosine and Pearson Correlation. The reason for this can be attributed to the sparse data. The SVD being a dimensionality reduction approach tackles this approach and produces better RMSE and MAE values.

3.3 Neural Collaborative Filtering

We have implemented Neural Collaborative Filtering[4], which incorporates Multi-layer perceptron to learn user-item interaction which can be seen in Figure [7]. User and Item feature vectors are passed as input to this model. Only the user and item identities are taken into account as input features. These are encoded using binarized one-hot binarization to create a sparse vector. It is important to note that by including content features to represent users and products, our strategy can be modified to address the cold-start issue.

The embedding layer is the layer above the input layer. The layer that turns the sparse representation into a dense vector is fully connected. This layer converts our high dimensional sparse vector to a low dimensional dense vector which in turn helps the model to learn much better. In the context of a latent factor model, the resulting user and item embeddings can be viewed as the underlying vector for each user and object. The neural collaborative filtering layers, a type of neural architecture, are then fed these embeddings. These layers use a multi-layer neural network technique to map the latent vectors to prediction scores. It is possible to tailor each layer in the neural CF layers to recognize particular latent structures of user-item interactions. The size of the final hidden layer, X, determines the model's capability.

. The output layer tries to predict the score $\hat{y}_{ui}$ which is user-item interaction. In our experiment, we try to predict the normalized rating of a movie i by a user u. Point-wise loss has been calculated between y_{ui} which is ground truth given as input and $\hat{y}_{ui}$.

$$\hat{y}_{ui} = f(P^T v_u^U, Q^T v_i^I | P, Q, \theta_f) \tag{4}$$

In equation 5, we can see that P and Q are the latent factor vectors that denote the user and items respectively. As part of pointwise

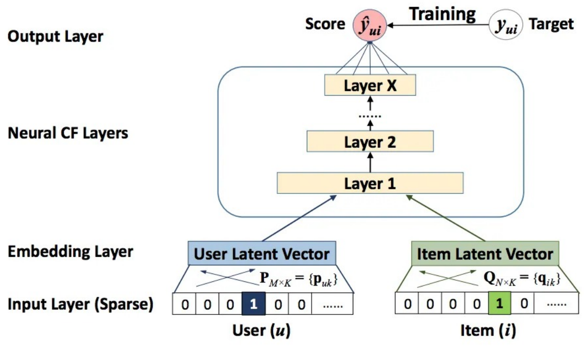


Figure 7: Neural Collaborative Filtering Framework

loss, we have incorporated Mean Absolute Error to learn the parameters of our model.

$$L = \frac{1}{n}\Sigma|\hat{y}_{ui} - y_{ui}| \tag{5}$$

The model can be customized to accommodate a wide range of users to items such as content-based, neighbor based, etc. As our dataset is small, we have focused on a collaborative filter-based model. In our model, we chose the size of the embedding layer as 50. We have included 2 hidden layers as part of our NCF framework. We also added a dropout of 0.1 in our model to reduce overfitting. We have used Adam optimizer for this experiment with its default hyperparameters with a learning rate of 1x10^{-3}.

After training the NCF model, we are getting average RMSE AND MAE for the rating prediction as follows:

$$\boldsymbol{RMSE} : \mathbf{0.52}$$
$$\boldsymbol{MAE} : \mathbf{0.49}$$

We also tried to make a recommendation system by recommending movies to users that they haven't watched before. We sorted our list of movies based on predicted ratings and chose the top 10 movies with the highest rating. For the top 10 recommendations of movies, we got Precision, Recall, F1 score, and NDCG score as follows:

$$\boldsymbol{Precision} : \mathbf{30.38\%}$$
$$\boldsymbol{Recall} : \mathbf{16.70\%}$$
$$\boldsymbol{F1\,Score} : \mathbf{21.55\%}$$
$$\boldsymbol{NDCG} : \mathbf{28.38\%}$$

. As we can see from the results, NCF has much better results than all the other algorithms for rating prediction and movie recommendation systems. One of the reasons is that NCF uses an embedding layer that converts sparse vectors to dense vectors that retain the original information.

4 EVALUATION CRITERIA

. There are several criteria that we can use to evaluate the goodness of the recommendation system. In this experiment, we have used Precision, Recall, F1 score, and NDCG to evaluate the goodness of the recommendation system. This project had a dual objective of constructing both a recommendation system and a rating predictor. To evaluate the performance of the rating predictors, we utilized mean absolute error (MAE) and root mean squared error (RMSE) as comparison metrics. Let's begin by examining these metrics.

4.1 Precision

The percentage of relevant instances among the derived instances can be used to define precision. The number of movies contained in both the test set and the recommended set is divided by the recommended set's size to determine precision.

$$Precision = \frac{tp}{tp + fp} = \frac{good\ movies\ recommended}{all\ recommendations} \tag{6}$$

4.2 Recall

The fraction of relevant instances that were recovered can be referred to as recall. The recall is determined by dividing the number of movies in the test set by the number of movies in the recommended set.

$$Recall = \frac{tp}{tp + fn} = \frac{good\ movies\ recommended}{all\ good\ movies} \tag{7}$$

4.3 F1 score

We can combine precision and recall using the F1 score because they are in direct competition with one another. The following is the F1 score formula:

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{8}$$

4.4 nDCG score

It is called normalized discounted cumulative gain. Equation of nDCG is given as below:

$$nDCG = \frac{DCG}{iDCG} \tag{9}$$

where *DCG* is discounted cumulative gain and *iDCG* is called ideal discounted cumulative gain. Both of the equations can be described as shown below:

$$DCG_p = \Sigma_{i=1}^{p} \frac{2^{rel} - 1}{log_2(i+1)} \tag{10}$$

$$iDCG_p = \Sigma_{i=1}^{rel} \frac{2^{rel} - 1}{log_2(i+1)} \tag{11}$$

5 RESULTS

Model	RMSE	MAE	Precision @10	Recall @10	F1 @10	nDCG @10
Cosine Similarity	1.98	2.12	11.27%	10.1%	10.65%	13.33%
Pearson Similarity	2.05	2.27	10.34%	9.57%	9.95%	11.78%
SVD	0.65	0.77	19.45%	11.20%	14.21%	19.20%
NCF	0.49	0.52	30.38%	16.70%	21.55%	28.38%

6 CONCLUSIONS AND FUTURE WORK

From the above results, we can say that NCF outperforms all the other algorithms in all the evaluation criteria. The main reason behind this is that NCF takes care of sparse data before it tries to learn use-item interaction using Multi-level perceptron. As it converts the sparse data into the low dimensional dense vector, it makes it easier for the model to train on this. In the table, we can see SVD gives competitive results with the NCF because SVD uses latent vectors which helps it to learn use-item interaction. However, SVD is not good when it comes to sparse dataset.

ACKNOWLEDGEMENT

We would like to express our gratitude to all those who have contributed to the completion of this technical report. First and foremost, we would like to thank our supervisor Prof. Yongfeng Zhang for providing us with the guidance, support, and valuable feedback throughout the project.

REFERENCES

[1] Dhiraj Vaibhav Bagul and Sunita Barve. 2021. A novel content-based recommendation approach based on LDA topic modeling for literature recommendation. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)*. 954–961. DOI: http://dx.doi.org/10.1109/ICICT50816.2021.9358561

[2] Zheng-Yi Chai, Ya-Lun Li, Ya-Min Han, and Si-Feng Zhu. 2019. Recommendation System Based on Singular Value Decomposition and Multi-Objective Immune Optimization. *IEEE Access* 7 (2019), 6060–6071. DOI: http://dx.doi.org/10.1109/ACCESS.2018.2842257

[3] Xin Guan, Chang-Tsun Li, and Yu Guan. 2017. Matrix Factorization With Rating Completion: An Enhanced SVD Model for Collaborative Filtering Recommender Systems. *IEEE Access* 5 (2017), 27668–27678. DOI: http://dx.doi.org/10.1109/ACCESS.2017.2772226

[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *CoRR* abs/1708.05031 (2017). arXiv:1708.05031 http://arxiv.org/abs/1708.05031