CS550: Comparison of Neural Collaborative Filtering with Individual Models

Daniel Ying dty16
Rutgers University
dty16@scarletmail.rutgers.edu

ABSTRACT

The growth of the internet and the rise of platforms like Google and Netflix resulted in not only an enormous amount of data and information, but also a trend of striving for better and more efficient tools that allow users to filter the massive information and obtain a possible item that users would want to see. Thus, recommendation systems are experimented and utilized in many contexts through multiple models. One of recommendation system's many context applications is the movie recommendation. Thus for this work, we will be focusing on implementing and analyzing the collaborative filtering.

Collaborative filtering is the technique used to filter the versatile data in order to obtain a possible item that a user might like, hence the name 'recommendation' system. Though there are multiple individual models explored and developed for the collaborative filtering, but the modeling of the key factor in collaborative filter is still resorted to the matrix factorization and apply an inner product on the latent features of users and movies. Thus the hybrid model known as the Neural Collaborative Filtering (NCF) framework was developed.

The NCF model is the concatenation of two models: the Generalized Matrix Factorization model (GMF) and the Multi-Layer Perceptron (MLP) model. Thus for this work, we will explore the popular individual collaborative filtering models, analyze the effect of the item-item and user-user similarities on such models, and develop and experiment with the neural collaborative filtering model. The data and measures we obtained from implementing the collaborative filtering models and the NCF shows that the hybrid algorithm indeed have a better performance than the individual non-hybrid models.

KEYWORDS

Collaborative Filtering, Matrix Factorization, Movie recommendation, neural networks, latent factors, individual models, hybrid model

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Due to the rise of internet, we have witnessed the era of versatile data where massive amount of information is available for users. Thus, the recommendation system have become an vital player in filtering the massive data and produce personalized recommendations that the user might want to see (thus the name 'recommendation' system). The recommendation system is able to accomplish this by modeling the data of user's past choices and interactions with items, ratings, reviews, and so on to obtain the user's possible preference. And One of the techniques in obtaining the model from the data is known as collaborative filtering (CF), which projects the users or items into a shared latent space vectors (we will talk more about latent space in description of neural collaborative filtering).

Among the other various collaborative filtering techniques is the Matrix Factorization. This technique has became popular due to the Netflix Prize. Many research has been devoted to this CF technique. But despite its success and effectiveness in collaborative filtering, the Matrix Factorization is not without flaws. The performance Matrix Factorization will be affected by by the inner product: as in user and item bias terms. Though the effect of the user and item on the Matrix Factorization can be tweaked by fine-tuning the inner product, but the problem with the inner product proves that we can design a model that has a dedicated interaction function in featuring the latent relation between users and the items. Thus we will be implementing Neural Collaborative Filtering (NCF).

The Neural Collaborative Filtering, though it would also resort to Matrix Factorization to combine the user and item latent features as a inner product, will be using deep neural networks for learning the interaction function from the data rather than using fine-tuning the user and item latent features to decrease the negative impact of the innter product. So Neural Collaborative Filtering will be a combination of two models, the Generalized Matrix Factorization and the Multi-Layer Perceptron.

Thus to explore the effect of the Neural Collaborative Filtering, we will implement the individual models that are not a combination of two models, which include: KNN series, Singular Value Decomposition, Baseline, SlopeOne, Non-negative Matrix Factorization, and CoClustering. The Data we will be using to train and test the models will be from the "grouplens movie review data-set," the ml-latest-small which has about 100K data.

1.1 About the Data:

The ml-latest-small data-set used for this work is from GroupLens, a research group in Department of Computer Science and Engineering at the University of Minnesota [5]. This data set is composed for 4 files that contained the data about ratings, movies, tags, and links. For this work, the ratings file will be the data we primarily use.

The ratings file contains 610 unique users and 9724 unique movies, with a total of 100836 ratings and 4 information (userId, movieId, rating, and timestamp) as one can see in figure 1.

```
ratings shape: (100836, 4)
first 5 ratings:
```

	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 1: Table from the Ratings.csv file.

For this work, the reviews from ratings dataset will be divided into two parts. 80% of the reviews made by the users will be randomly selected to be the training set, while the other 20% of the reviews made by the users will be defined at the testing set.

2 RELATED WORK

2.1 Neural Collaborative Filtering[6]

This paper explores the application of applying neural network, the deep neural networks, into the model. The objective of the paper focuses on the problem of inner product, in which the latent features of the users and items will affect the performance of the models including the Matrix Factorization model. The paper stated that even though the feature effect can be tuned out by various techniques but this shows that there is room for improvement through the design of the model targeting the latent features of the users and items. Thus we proposes the use of adding deep learning layers to the collaborative filtering model, results in a model called Neural Collaborative Filtering. This model is the combination of Generalized Matrix Factorization and the Multi-Layer Perceptron. Thus the paper applied multiple experiments on two real-world data sets to show and observe how the proposed Neural Collaborative Filtering model performed compared to the current models. The conclusion of the paper show that the Neural Collaborative Filtering model does succeed in improved recommendation performance through using the deeper neural networks of NCF.

3 PROBLEM FORMALIZATION

3.1 Baseline[4]

The Baseline Algorithm, used top predict the baseline estimation for a specific user and item, has the formula of:

$$\hat{r}_{ui} = \mu + b_i + b_u \tag{1}$$

The r is the prediction rating based on the bias deviation of the item and user. The μ represents the average rating while the b_u, b_i represents the bias deviation of the user and the item. The training data set would be passed into the Baseline Algorithm to 'train' and obtain a model with which we can use to predict an estimated prediction rating with the test data set as the parameter.

The advantages of the Baseline Algorithm is it is fairly easy and simple to implement while are the same time obtain reasonably good results and performance. This algorithm allows us to put more complex models into context in terms of improvement in accuracy by comparing the Baseline Algorithm with the complex model's results. While the disadvantage of the Baseline Algorithm is, though it is simple and quick to be implemented, the results is not the best and its performance can be improved when we compare its results to those of other models and especially complex models that involve deep neural networks.

3.2 Singular Value Decomposition[4]

The Singular Value Decomposition Algorithm also known as SVD, is used as the collabortaive filtering to reduce the number of features in the data set by reducing the dimensions of the data's matrix. The mathematical formula for SVD is:

$$A = P\Sigma Q^T \tag{2}$$

The original data matrix A is decomposed into P, Σ, and Q^T. The data would have dimensions of mxn, the P would have dimensions of mxm, Σ would have mxn, while the Q^T would have dimensions of nxn. Thus the relation of the matrices can be expressed as the rating prediction of:

$$\hat{r}_{ui} = \mu + b_i + b_u + (q_i)^T p_u \tag{3}$$

The variables that appeared in the Baseline Algorithm represents the same features of the user, items and ratings. The extra variable p and q represents the concepts of user and item, where q_i represents the movie concept and the p_u represents the user concept.

By reducing the dimensions of the data matrices, the advantages of implementing the SVD includes: simplifying the complexity of the data, remove the noise the data might have, and usually improves the algorithm results, especially if we compare SVD's performance to the performance of the Baseline Algorithm. But the disadvantages of the SVD algorithm also comes from reducing the matrices. The disadvantages of the algorithm includes making the transformed data more difficult to understand than the original data.

3.3 Non-negative Matrix Factorization[4]

The Non-negative Matrix Factorization is very similar to the SVD algorithm. The Non-negative MAtrix Factorization is also a collaborative filtering that has the formula:

$$\hat{r}_{ui} = q_i^T p_u \tag{4}$$

For this algorithm's formula, without the bias deviations of the user and items like in SVD, we can see that the user and item concept factors are kept as a posibtive, thus the prediction rating for

the Non-negative Matrix Factorization would be non-negative (as the name implies).

The advantage of the NMF algorithm would be: NMF encourages the imputation of missing signal, enforces sparsity, ensure that factors from user and item would not cancel one another out. The disadvantage of the NMF algorithm would be there exists no unique global minimum for the NMF for the algorithm can only guarantee the convergence to a local minimum. Thus even if the same NMF algorithm was implemented, if we change the parameters slightly, the algorithm would produce different NMF results.

3.4 Slope One[4]

The Slope One algorithm is a simple collaborative filtering algorithm that is based on the linear lreation between the user and the movie. The Slope One algorithm thus, would use the linear regression model to predict the rating. So its formula would be defined as:

$$\hat{r}_{ui} = \mu_u + \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} dev(i, j) \tag{5}$$

The prediction rating would be r_{ui} while the $R_i(u)$ is the set of relevant items as in the set of items j rated by the user u which the user u would also have at least one common user i. The dev(i,j) is defined as the average different between the ratings of user i and those made by user j.

$$dev(i, j) = \frac{1}{|U_{ij}} \sum_{u \in U_{ij}} r_{ui} - r_{uj} \tag{6}$$

The advantage of the Slope One Algorithm is it is easy to implement, new ratings can will change the rating predictions quickly, efficient in runtime since the item pairs are already stored (needs a lot of memory), little user feedback is needed, and it is reasonably accurate (better than Baseline Algorithm).

One of the main disadvantage of the Slope One Algorithm is that it is memory intensive. The amount of memory needed to store all the computed differences between the item and user pairs is equal to the square of the total number of items. Thus is would need four times the memory needed to store the items.

3.5 Co-Clustering[4]

The Co-Clustering Algorithm is a collaborative filtering algorithm that is based on co-clustering. This algorithm assigns users and items to clusters of C_u, C_i, C_{ui}. And clusters are assigned using a straightforward optimization method, thus the formula for the Co-Clustering Algorithm would be:

$$\hat{r}_{ui} = avg(C_{ui}) + (\mu_u - avg(C_u)) + (\mu_i - avg(C_i)) \tag{7}$$

The average of each assigned co-cluster for ui (user-item), u (user), and i (item) If the user is unknown, the algorithm would have a prediction rating that equals to μ_i. If the item is unknown instead, the prediction rating would be μ_u. Lastly if both the user and item are unknown then the prediction rating would be μ.

The advantages of Co-Clustering Algorithm are: when the algorithm clusters the user and items, it will decrease the dimension of clustering, while appropriately measuring the distance between the datapoints, and when it is mining more useful information, the algorithm can get the corresponding information in the clusters. Thus the Co-Clustering Algorithm allows us to find the clusters in a subset of features.

The disadvantage of the Co-Clustering Algorithm is it inability to find a scaling pattern in its co-clustering analysis. This is because groups of users and items are showing similar patterns with different scales are also the meaningful clusters we are aiming to identify.

3.6 K Nearest Neighbors Algorithm[4]

K Nearest Neighbors Algorithm is used to find clusters of similar items. This algorithm forms clusters of the datapoints from the dataset. Once the clusters are formed,the kNN algorithm can predict the rating of a movie as the average of the movies with similar ratings. The kNN algorithm relies on the datapoint similarity to make its rating predictions, thus it calculates the distance between the target datapoint to the neighboring datapoints around the target (number of neighboring datapoints are numbered to k thus the name of kNN) and returns the best k neighbors as the predicted rating for the target movie. How the kNN algorithm decides which neighboring datapoints are considered as the k best neighbors are what branches the Basic kNN into other varieties of kNN, which we will mention in the next section.

The advantages of kNN include: there is no training period, new data can be added without impacting the overall accuracy of the algorithm, and the kNN algorithm is easy to implement.The disadvantage of kNN include: it does not work well with large datasets, does not work well with high dimensions (large number of dimensions), it need feature scaling, and it is sensitive to noisy data, missing data and outliers.

Next we will talk about the varieties of kNN algorithms and the metrics that each kNN algorithm have.

3.7 Types of kNN [4]

3.7.1 Basic kNN. The kNN Basic Algorithm has the formula of:

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} sim(i, j) * r_{uj}}{\sum_{j \in N_u^k(i)} sim(i, j)} \tag{8}$$

The kNN Basic Algorithm takes into account the maximum k number of neighboring datapoints and have the similarity metrics as the algorithms parameters.

3.7.2 kNN with Means. The kNN with Means Algorithm has the formula of:

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} sim(i, j) * (r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} sim(i, j)} \tag{9}$$

The kNN with Means Algorithm is similar to the basic kNN algorithm but the different between the two is: the kNN with Means Algorithm takes into account the mean ratings of each user.

3.7.3 kNN with Z-Score. The kNN with Z-Score has the formula of:

$$\hat{r}_{ui} = \mu_i + \sigma_i \frac{\sum_{j \in N_u^k(i)} sim(i,j) * (r_{uj} - \mu_j)/\sigma_j}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{10}$$

Compared with kNN with Means, the kNN with Z-Score takes into account the z normalization of each users instead of the means between users.

3.7.4 kNN Baseline. The kNN Baseline has the formula of:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} sim(i,j) * (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{11}$$

Compared with kNN with Means and kNN with Z-Score, the kNN Baseline takes into account of the baseline rating instead.

3.7.5 kNN Metrics. The kNN possessed Metrics that change how the kNN algorithm would measure the similarity between datapoints. Such metrics are:

- Cosine: measure similarity between 2 non-zero vectors
- Mean Square Difference (MSD): the mean square similarity is defined as inversely proportional to the difference of users' mean square difference
- Pearson: mean-centered cosine similarity
- Pearson Baseline: baseline-centered cosine similarity

3.7.6 User-User vs. Item-Item Similarity. The collaborative filtering can be differentiated into user-user and item-item based similarity collaborative filtering. If the collaborative filtering is user-user similarity then the neighbors of user i is defined by users who have given similar ratings to the items also rated by user i. The item-item similarity on the other hand, is item j's neighbors are also items that have been rated users that also rated item i. The difference in similarities will have a significant effect on the performance of the models, thus we shall implement both the user-user similarities and the item-item similarities to observe the effect on the two similarities.

4 THE PROPOSED MODEL: NCF

Our proposed model is Neural Collaborative Filtering (NCF). This algorithm model is a deep neural network model that combines two different algorithms: the Generalized Matrix Factorization algorithm and the Multi-Layer Perceptron algorithm. One important feature of the Neural Collaborative Filtering algorithm is its ability to allow users to tun the user and item embeddings separately, thus allowing flexibility and implementation the previously mentioned algorithms did not possess.

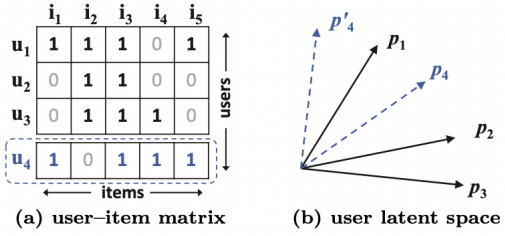
Thus to understand the structure and implementation of Neural Collaborative Filtering algorithm we will talk about the Matrix Factorization, General Matrix Factorization, and Multi-Layer Perceptron.

4.1 Limitation of Matrix Factorization [6]

The Matrix Factorization (MF) algorithm pairs the user and item with real-valued vectors that represent the latent features of each. Thus this can be represented as:

$$\hat{y}_{ui} = f(u, i|p_u, q_i) = p_u^T p_i = \sum_{k=1}^{K} p_{uk} q_{ik} \tag{12}$$

The p_u and q_i represents the latent feature vectors for user u and item i, and the y_{ui} represents the interaction of the user and item latent vectors which is the inner product of the p and q. While K represents the dimension of the inner product. Thus, since MF algorithm maps the user and item to the same latent space, the similarity of the two would be measured with the inner product, as can be seen in figure 1.



(a) user–item matrix (b) user latent space

Figure 2: The figure shows the MF's limitation. (a) shows that u_4 is most similar to u_1, then u_3 is second most similar, lastly u_2 least most similar. But looking at latent space (b), it shows that by putting p_4 closest to p_1, p_4 would be closer than p_2 instead of p_3 thus showing ranking loss in the transition from matrix to latent space.[6]

The figure shows the possible limitation of the MF algorithm. The transition from the user-item matrix to the the lower dimension user latent space may lead to ranking loss, this is caused by the use of simple and fixed inner product to estimate the user-item interactions.

Thus to improve the decreased performance due to the transition of the user-item matrix to the user latent space, our Neural Collaborative Filtering Algorithm would generalize the Matrix Factorization. Thus we will next talk about the Neural Collaborative Filtering Algorithm and the two models that compose the NCF algorithm: Generalized Matrix Factorization and Multi-Layer Perceptron.

4.2 Neural Collaborative Filtering: Generalized Matrix Factorization [6]

As stated before, the Neural Collaborative Filtering algorithm is composed of Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP). At first glace, one would ask why is the MF algorithm called GMF instead in the NCF model? What is the difference between the MF and the GMF model? The application of the GMF in the NCF can be represented as the mapping of GMF as the formula:

$$p_u = P^T v_u^U, p_i = Q^T v_i^I$$
$$dot(p_u, q_i) = p_u \odot q_i \tag{13}$$

The $\odot$ represents the element-wise product of vectors, the formula can then be deduced as:

$$\hat{y}_{ui} = a_{out}(h^T(p \odot q_i)) \tag{14}$$

The a_{out} and h represents the activation function and the weights of the output layer as can be see in figure 3.

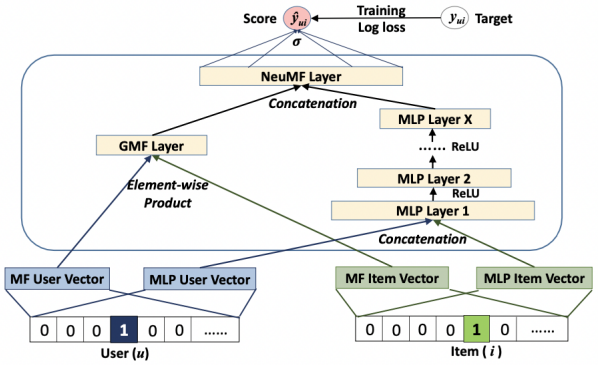


Figure 3: Neural matrix factorization model

Figure 3: [6]

So in the NCF architecture, the MF would be updated into the GMF through the application of the a_{out} and the h variable in the formula above. This is possible because the h is learned from the data without the uniform constraint making it into a variable of MF that allows it to be the factor of importance for the latent dimensions. While a_{out} on the other hand would be used to generalize the MF into a non-linear setting allowing the transition from the user-item matrix into the latent space to be more expressive than the original linear MF.

Thus for the NCF architecture, GMF is the generalized version of MG to which it applies the a_out from the sigmoid function and learns the h variable with the log loss from the data.

4.3 Multi-Layer Perceptron [6]

The second model that makes up the NCF is the Multi-Layer Perceptron (MLP). The MLP contains the deep neural network of NCF. As can be seen in figure 3, the MLP framework contains the hidden layers that learn the user and item latent features and similarities from the concatenated vectors from user u and item i. Thus the MLP model gives us high flexibility and non-linearity to learn the interactions of p_u, q_i. The MLP model is more complicated and is the core of the NCF algorithm since the GMF model only uses a fixed element-wise product on the user u and item i data set.

We can apply the MLP as the formula:

$$\begin{aligned} z_q &= dot(p_u, q_i) = vector[p_u, q_i] \\ dot_2(z_1) &= a_2(W_2^T z_1 + b_2) \\ &\dots \\ dot_L(z_L) &= a_L(W_L^T z_{L-1} + b_L) \\ \hat{y}_{ui} &= \sigma(h^T dot_L(z_{L-1})) \end{aligned} \tag{15}$$

The W_x, b_x, a_x represents the weight matrix, bias vector, and the activation function for the layer x of the MLP layer. It is important to note that there are three activation functions considered for the MLP layer and they are:

- sigmoid
- tanh (hyperbolic tangent)
- ReLU

We have chosen ReLU to be the activation function of the MLP. This is because sigmoid suffers from saturation since its output is either 0 or 1. And although tanh does improve the problem of saturation in sigmoid, but the problem still persists in tanh. While, ReLU was chosen because it encourages sparse activations thus making it suitable for sparse data. Thus by using ReLU, the MLP model is less likely to overfit.

The MLP would form the vector of the user and item embeddings by inputting them into the input layer. The hidden layers would process and learn from the embeddings and forward the processed embeddings to the next layer until the inputs have reached the last layer, which is the fusion of the MLP with the GMF results.

4.4 NCF: Fuse GMF and MLP [6]

After defining the MLP and GMF with the models' structure and formula by inputting the user and item data, we will proceed in fusing the two models to complete the Neural Collaborative Filtering framework (as can be seen in figure 3). As stated from before, NCF provides a flexibility due to the fact we can separately tune the models GMF and MLP for the two models learn the user and item embeddings separately. After doing so, we will concatenate the two model's last layer to obtain the NeuMF layer (Neural Matrix Factorization) in figure 3.

The concatenation of the two models can be represented as the formula:

$$\hat{y}_{ui} = \sigma(h^T a(p_u \odot q_i + W * vector[p_u, q_i]) + b)) \tag{16}$$

And to deduce this formula into step by step:

$$\begin{aligned} GMF &= p_u^G \odot q_i^G \\ MLP &= a_L(W_L^T(a_{L-1}(\dots a_2(W_2^T * vector[p_u^M, q_i^M] + b_2)\dots)) + b_L) \\ \hat{y}_{ui} &= \sigma(h^T * vector[GMF, MLP]) \end{aligned} \tag{17}$$

The p_u^G, p_u^M represent the user embeddings in MLP and GMF, while the q_i^G, q_i^M represent the item embeddings in MLP and GMF. The NCF model combines the linearity of GMF and the non-linearity

of MLP's deep neural network when modeling and learning the user-item latent features and similarities, thus this final layer is called the Neural Matrix Factorization (NeuMF), as seen in figure 3.

Thus the by fusing GMF, MLP, NCF's last layer (NeuMF) would be formed, thus completing the architecture of NCF, allowing us make prediction ratings from the test data set.

5 EXPERIMENTS

Before we can show the results from out individual models and the NCF, we have to introduce our measures that we use to observe the differences between the different models.

5.1 Measures

5.1.1 Root Mean Square Error [2]. RMSE is the square root of the variance of the residuals. The formula is:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{18}$$

This formula is the absolute fit of the model's predicted rating to the actual rating. Thus the closer the predicted and actual ratings are, the smaller the root mean square error is.

5.1.2 Mean Absolute Error [1]. The MAE measures the average of the errors in the set of prediction ratings. The formula is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \tag{19}$$

MAE is similar to RMSE, but it is simpler and easier to interpret than the later. The MAE is the average absolute distance between X and Y.

5.1.3 Precision.

$$Precision = \frac{|\text{Recommended relevant items}|}{|Recommendeditems|} \tag{20}$$

Precision measures the ability of the model to predict items that are both relevant and recommended.

5.1.4 Recall.

$$Recall = \frac{—\text{Recommend relevant items}—}{—\text{Relevant items}—} \tag{21}$$

Recall measures the ability of the model to predict items that are both recommended and relevant.

5.1.5 F-Measure.

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{22}$$

F-Measure measures the model's accuracy based on the precision and recall measures.

5.1.6 Normalized Discounted Cumulative Gain[3].

$$\begin{aligned} DCG_p &= \sum_{i=1}^{p}\frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^{p}\frac{rel_i}{log_2(i+1)} \\ IDCG_p &= \sum_{i=1}^{|REL_p|}\frac{rel_i}{\log_2(i+1)} \\ nDCG_p &= \frac{DCG_p}{IDCG_p} \end{aligned} \tag{23}$$

The NDCG normalizes the discounted cumulative gain, sorts all relevant documents by relative relevance, and produce the max possible DCG, as in the ideal value for the position in p.

5.2 Comparison of Models RMSE and MAE

Looking at figure 4 Table, we can see that among the individual models, kNN Baseline with metrics of Pearson Baseline based on item-item similarities had the be MAE rating. And for the next 3 best MAE values are all kNN Baseline algorithm. While the worst performing model in the figure 4 Table in terms of MAE value is the kNN Basic with cosine metrics based on the item-item similarities.

Looking at figure 5 Table, the model that had the best value is still the kNN Baseline with Pearson Baseline metrics based on item-item similarities model. Although the second best RMSE value is still the kNN Baseline model, but the third and fourth best RMSE models are actually the Baseline Only model and the SVD model. While the worst performing model in the figure 5 Table in terms of the RMSE value is the kNN with cosine metrics based on the item-item similarities.

Looking at figure 6 Table, the worst performing model in terms of both MAE and RMSE in all the models implemented is the GMF model. This was expected, since although the GMF model had implemented a_{out} and h value to generalize and reduce the negative impact of the transition from the data matrices to the linear latent space, the loss of rank is inevitable. Thus the performance of the GMF is the worst is not unexpected. While for the NCF, the performance of our deep neural network model is average in comparison with the individual models implemented in the figure Tables 4 and 5.

5.3 Comparison of the 4 Measures

Unlike the results we observed in the RMSE and MAE tables for the individual models, if we look at the 4 measures from figure 7 Table (Recall, Precision, F-Measure, NDCG), the best performing model is the kNN Baseline. And if we take a closer look at the kNN Baseline results, we can see that the item-item similarity has better performance than the user-user similarity, thus supporting the statement that item-item similarity is better than user-user similarity for collaborative filtering models.

Interestingly, if we turn our attention to figure 8 Table, the 4 measure of GMF is significantly lower than the models in the figure 7 Table. The worse performance in the 4 measures of the GMF proves that the negative effect of the inner product on the performance of the GMF model still persists despite the application of the variables

$a_{out}, h.$

As described in the previous sections, NCF is the fusion of the two models GMF and MLP. The deep neural network MLP can be seen as an important component of the NCF. The effect of the MLP can be seen in the prediction results we get from the NCF model from the NeuMF layer, obtained by fusing the previously mentioned GMF with the MLP. Looking at figure 8 Table again, we can see that the recall, precision and f-measure results of the NCF are all significantly higher than the models in the figure 7 Table.

This shows that despite the NCF taking much more time to be trained and predict the testing data set, the results and performance of the NCF are better than the individual models in the figures 4, 5, and 7 Tables. Thus our work is a success. We have proven that the inner product does have a negative effect on the models' performance and we have proven that with a designed model like deep neural networks, the negative effect of the inner product can be minimized thus producing better results.

6 CONCLUSIONS AND FUTURE WORK

Matrix Factorization has become one of the most common collaborative filtering techniques implemented in recommendation systems. But this useful algorithm has the problem with inner product, where the transition from the data matrix to the linear latent space is faced with the loss of rank. Though we can fine tune out the negative effect of the inner product, but the issue the MF possess shows that there is room of improvement. Thus the Neural Collaborative Filtering [6] is proposed.

By separately modeling the Generalized Matrix Factorization and Multi-Layer Perceptron (deep neural network) with the embeddings and fusing them in the NeuMF layer, we will obtain the NCF model. And from the predicted ratings made by the trained NCF model, the recall, precision, and the f-measure show that the NCF does have better performance than the Matrix Factorization and other individual models implemented in this work.

For the future work, we can increase the parameters of the models. This work focuses on the userId and movieId from the Ratings.csv file. We did not implement the genres data in the csv file. Also, we can try to implement other hybrid algorithms and observe how such models perform in comparison with the indivudal models and the NCF DNN model.

It is also observed, by increasing the dataset size or dimensions, models such as kNN would have difficulty in obtaining good performance due to the design of the model. Our dataset is the 100K data set. There exists the 1M data set and the much larger Complete Latest Dataset in the GroupLens database. The mentioned future works would be interesting to implement and would be the next extension of this work.

ACKNOWLEDGEMENT

We would like to thank Professor Yongfeng Zhang and Teaching Assistance Zhiqing Hong for their guidance, support, and providing us the necessary knowledge, resource, and help in completing this work.

REFERENCES

[1] Wikipedia Contributors. 2021. Mean Absolute Error. *https://en.wikipedia.org/wiki/Mean$_a$bsolute$_e$rror*(2021).
[2] Wikipedia Contributors. 2021. Root Mean Square Deviation. *https://en.wikipedia.org/wiki/Root-mean-square$_d$eviation*(2021).
[3] Wikipedia Contributors. 2022. Discounted Cumulative Gain. *https://en.wikipedia.org/wiki/Discounted$_c$umulative$_g$ain*(2022).
[4] Surprise Library Developers. 2015. Surprise Documentation. *https://surprise.readthedocs.io/en/stable/index.html9* (2015).
[5] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19* (2015).
[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *arXiv: 1708.0503lv2 [cs.IR]* (2017).

KNNBaseline Item Pearline	0.858280224151538
KNNBaseline Item MSD	0.872836190393700
BaselineOnly	0.876689030302501
SVD	0.879544889916836
KNNBaseline User MSD	0.88158399508237
KNNBaseline Item Pear	0.883541800486748
KNNBaseline User Pearline	0.884051986277553
KNNBaseline User Cos	0.884943109330249
KNNBaseline User Pear	0.885017054928852
KNNBaseline Item Cos	0.903377924247580
KNNWithMeans User	0.903482744129088
KNNWith Means Item	0.903482744129088
KNNZ User	0.903482744129088
KNNZ Item	0.903482744129088
SlopeOne	0.906643008429340
KNNBasic Item MSD	0.914964601352633
NMF	0.929199477676719
CoClustering	0.950168887214654
KNNBasic User MSD	0.955627428794643
KNNBasic User Cos	0.979848184079456
KNNBasic Item Cos	0.995619956790904

Figure 5: Table for the RMSE of the individual models, ordered in ascending order.

NCF_MAE_RSME_table

	Algorithm	MAE	RSME
0	NCF	0.679384940734982	0.9022068009032480
1	GMF	0.7751338284519320	1.0949647014682100

Figure 6: Table for the RMSE and MAE of NCF and GMF.

	Algorithm	Recall	Precision	F-Measure	NDCG
0	svd	0.631081	0.746740	0.684056	0.950120
1	slopeone	0.632387	0.747549	0.685163	0.946630
2	nmf	0.604784	0.718138	0.656605	0.943497
3	knnz-user	0.637244	0.749144	0.688678	0.941094
4	knnz-item	0.637244	0.749144	0.688678	0.941094
5	knnmean-user	0.637244	0.749144	0.688678	0.941094
6	knnmean-item	0.637244	0.749144	0.688678	0.941094
7	knnbasic-user	0.662519	0.797699	0.723852	0.948319
8	knnbasic-item	0.622308	0.730851	0.672226	0.941343
9	knnbaseline-user	0.645371	0.765710	0.700409	0.945723
10	knnbaseline-item	0.634790	0.750635	0.687869	0.951757
11	Coclustering	0.604260	0.716617	0.655660	0.943728
12	baseline	0.631700	0.751230	0.686299	0.951494

Figure 7: Table for the 4 measures of the individual models.

NCF 4 measures

	Algorithm	Recall	Precision	F-Measure	NDCG
0	Neural Collaborative Filtering	0.7325553467000840	0.8661432226399330	0.7937679571638840	0.9389520705161180
1	Generalized Matrix Factorization	0.5960095551378440	0.6961244256474520	0.6421885275373240	0.9475637658465470

Figure 8: Table for the 4 measures of the NCF.

KNNBaseline Item Pearline	0.6544214791312891
KNNBaseline Item MSD	0.6686846948739565
KNNBaseline User Pearline	0.6726476067823245
KNNBaseline User MSD	0.6728208025621423
SVD	0.6748513010089627
KNNBaseline User Pear	0.6758026468389403
KNNBaseline User Cos	0.6761500838166293
BaselineOnly	0.6763669135845803
KNNBaseline Item Pear	0.6791822334366855
KNNWithMeans User	0.6886208888687823
KNNWith Means Item	0.6886208888687823
KNNZ User	0.6886208888687823
KNNZ Item	0.6886208888687823
SlopeOne	0.6927328710255556
KNNBaseline Item Cos	0.6960748100521774
KNNBasic Item MSD	0.7028014662901718
NMF	0.7098917623418507
KNNBasic User MSD	0.7305797074310052
CoClustering	0.7350673342266314
KNNBasic User Cos	0.7526526548517168
KNNBasic Item Cos	0.7742228892538924

Figure 4: Table for the MAE of the individual models, ordered in ascending order.