Evaluation of Neural Collaborative Filtering (NeuMF) on modified MovieLens dataset

Apostolos Giannousas 02992, Athanasios Kastoras 03101, Christos Kyriakidis 03029,
Ioanna-Diamantoula Tzalla 02942, Spyridon Barmpakos 02937
dept. of Electrical and Computer Engineering, University of Thessaly
Volos, Greece
{apogiannousas, akastoras, ckyriakidis, itzalla, sbarmpakos}@uth.gr

***Abstract*—Using a subset of the MovieLens dataset, this project assesses the performance of the Neural Matrix Factorization (NeuMF) model for recommendation systems. To improve accuracy, this model combines Generalized Matrix Factorization (GMF) and Multi-Layer perceptron (MLP) methods. The evaluation is based on hit rate (HR) and normalized discounted cumulative gain (NDCG) metrics. Experiments are carried out to investigate the effect of the number of predictions and negative samples used for training. The results show that the NeuMF model performs well, but the reduced number of ratings per user in the smaller dataset used in this project affects its accuracy. This study looks into the factors that influence the performance of recommendation systems, emphasizing the significance of per-user average information.**

I. Introduction

A. Recommendation Systems

Over the last few decades, we have seen a massive increase in the usage of web services such as online shopping, streaming services, and social media in our everyday lives. With the exponential growth of online content provided by these services, there is an increased need for effective specialized recommendation systems. These systems, use factors such as demographic information, and search history to provide users with personalized suggestions and assist them in discovering new products and services that they might not have interacted with before.

Recommendation systems most commonly implement either collaborative filtering or content-based filtering, while there are others that use a combination of two or more strategies in what is known as a hybrid model, to benefit from their complementary advantages. Collaborative filtering is based on the idea that users who had similar preferences in the past will have similar preferences in the future. It utilizes similarities between multiple agents, viewpoints, and data sources to provide recommendations to users. On the other hand, content-based strategies use the features of each item to recommend similar items to the user's preferences, based on both implicit and explicit feedback.

B. Collaborative Filtering

As mentioned above, collaborative filtering is a method of making predictions about a certain type of item that a user might be interested in, based on their general preferences as well as the opinion of similar users on that specific item. Note that similar users are those who have very similar preferences on a certain variety of items to the target user.

This recommendation method can be separated into two major categories, namely neighborhood-based and model-based collaborative filtering. The former can be tackled by a user-based or item-based approach. According to the user-based approach, we can recommend a number of items that a given user might like by using the opinions of a number of similar users (i.e. users that have similar preferences to the given one). On the contrary, in the item-based approach, we can predict any missing ratings for a given item based on a group of items that have received similar ratings from each user with the given item.

The downside of those techniques is that they use a rather sparse user-item matrix and that their predictions are quite obvious since they don't exploit any hidden relations between the users and the items. Matrix Factorization (MF), which is a model-based method, can remedy those issues since it decomposes the user-item matrix into the product of two smaller matrices (or more in the general case), namely the user latent and item latent matrices. The common dimension of those two new matrices is the number of latent factors and it is optimized during training since it makes the model capable of making more sophisticated predictions based on hidden features.

II. NeuMF Model

The collaborative filtering technique connects the user latent and the item latent using vanilla matrix multiplication. This is a linear relationship that may not fully model the interaction of users and items. In [2] the proposed method is Generalized Matrix Factorization (GMF) which extends the functionality of the vanilla MF and lets it capture the relationship more accurately. It does that by using a non-linear output function and a learnable vector. The GMF formula is the following:

$$\hat{y}_{ui} = a_{out}(\boldsymbol{h}^T(\boldsymbol{p}_u \odot \boldsymbol{q}_i))$$

where $\odot$ denotes the element-wise product of two vectors, $\boldsymbol{p_i}$ is the user latent vector and $\boldsymbol{q_i}$ the item latent vector. The GMF model is indeed a generalized version of simple Matrix Factorization, which can be proved intuitively by setting a_{out} to the identity function and $\boldsymbol{h}^T$ to a uniform vector of ones.

Additionally, He et al. [2] propose the use of an MLP to perform such mapping, since a series of non-linear layers can help to model more complex relationships.

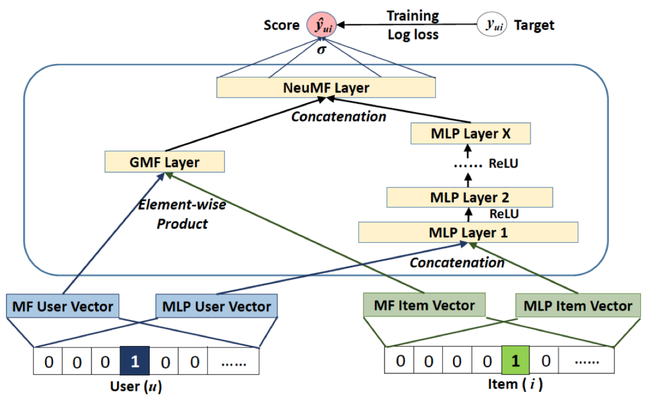

Fig. 1. Neural Matrix Factorization Model

The optimal model which was finally proposed was a combination of both methods as shown in figure 1. In essence, the outputs of both the MLP and GMF are computed and then concatenated into a longer vector, which is fed into a final neuron that squeezes the values into a smaller probability vector.

To optimize the model, He et al. [2] propose pretraining the GMF and the MLP parts separately and use their weights to initialize the NeuMF. By doing that, training the NeuMF requires only training a final fully connected neuron that takes as input the concatenated outputs of GMF and MLP and produces the final output of the NeuMF framework.

III. Dataset

A. Description

One of the datasets used in the original paper is the MovieLens Dataset [1], used often for the evaluation of recommendation systems. The specific dataset contains 1.000.000 ratings out of 6040 users, with a minimum count of 20 ratings for each user and an average of 165 ratings per user. Each line contains the user ID, the item with which they interacted, the rating and the timestamp of the last interaction. The model makes use of implicit data, therefore the rating for each movie is discarded and a record is kept for each interaction between each user and item encoded with 0's and 1's.

The dataset given to us is much smaller with 100.000 ratings of 943 users, about 106 ratings per user. The training dataset contains records of the interactions of each user with a specific item and the timestamp for each one of them. In order to create the negative dataset used for testing, we keep for each user the item with which he/she most recently interacted and 99 items randomly sampled from the ones with which he didn't interact.

B. Statistical Analysis

The reduced rate of ratings per user is likely to decrease the performance of the model in this dataset. This factor plays a more important role than the size of the dataset, since less information is given for each user on average which will lead to worse prediction accuracy. In figure 2, we observe that the probability distributions of the users' number of ratings for both datasets are similar, and therefore the average number of ratings per user is enough to indicate that the given dataset provides less information about the users than the original MovieLens dataset.

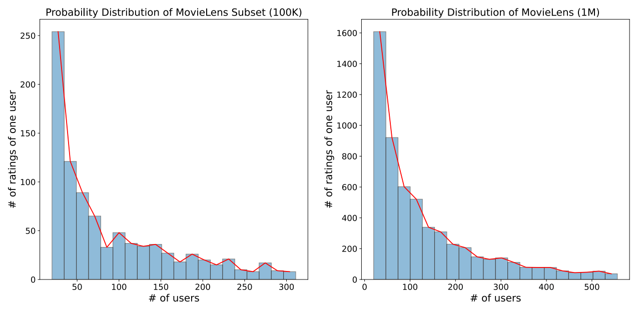

Fig. 2. Probability distribution for the number of ratings of a user for the given dataset (left) and the dataset used in [2] (right). The 10% of the users with the highest number of ratings are considered outliers and are being discarded to ensure the reliability of the analysis.

Table I presents the mean number of ratings per user for both datasets having removed only the top 10% of the users with the most ratings. Users with few ratings were not removed since all users have at least 20 ratings and it was observed that small values around the threshold of 20 were dominant and could not be characterized as outliers.

TABLE I
AVERAGE NUMBER OF RATINGS PER USER

Outliers Removed	**MovieLens (1M)**	**MovieLens Sample (100K)**
0%	165 Ratings/User	106 Ratings/User
Top 10%	114 Ratings/User	81 Ratings/User

IV. Accuracy Metrics

To evaluate the performance of the model, two metrics were used; hit rate (HR) and normalized discounted cumulative gain (NDCG). HR counts how many of the items in the recommendation list are correct. In our case, we only leave one positive sample out of the training phase and use it for evaluation, therefore in HR we just have to check if this sample is in the recommendation list. The final hit rate is calculated as the number of users whose evaluation rating (i.e. the rating which was left out of training) is in the recommendation list divided by the total number of users. Meanwhile, with the use of the NDCG metric, we give higher scores to hits at the top ranks. The NDCG formula is:

$$NDCG@K = Z_K \sum_{i=1}^{K} \frac{2^{r_i - 1}}{log_2(i+1)}$$

TABLE II
PERFORMANCE OF NEUMF FOR VARIOUS TOP-K VALUES

Top-K	1	2	3	4	5	6	7	8	9	10
HR@K	0.168	0.276	0.360	0.430	0.489	0.536	0.571	0.609	0.638	**0.667**
NDCG@K	0.168	0.235	0.276	0.305	0.325	0.343	0.359	0.362	0.372	**0.382**

TABLE III
PERFORMANCE OF NEUMF FOR VARIOUS NUMBERS OF NEGATIVES

Negatives	1	2	3	4	5	6	7	8	9	10
HR@10	0.650	0.659	**0.669**	0.663	0.659	0.667	0.665	0.663	0.661	0.659
NDCG@10	0.369	0.379	0.382	**0.386**	0.382	0.386	0.384	0.385	0.382	0.381

where Z_K is the normalizer to ensure the perfect ranking has a value of 1. r_i is the graded relevance of the item at position i. In this implementation, a binary relevance is used: $r_i = 1$ if the item is in the test set, and 0 otherwise.

V. Experimental Evaluation

To evaluate our model on the given dataset, we want to observe the performance using both HR and NDCG in respect to two variables, k (i.e. the number of predictions we list, in which we search for the correct item when evaluating the accuracy metrics) and the number of negative samples per positive sample, that are used for training. Therefore we will perform the following 4 experiments:

1) **HR@K vs K**: Calculate the Top-K HR for k from 1 to 10 with a stride of 1
2) **NDCG@K vs K**: Calculate the Top-K NDCG for k from 1 to 10 with a stride of 1
3) **HR@10 vs Num of Negatives**: Calculate the HR@10 changing the number of negative samples used for training for each positive sample from 1 to 10 with a stride of 1
4) **NDCG@10 vs Num of Negatives**: Calculate the NDCG@10 changing the number of negative samples used for training for each positive sample from 1 to 10 with a stride of 1

To reduce statistical errors we repeat each experiment 5 times, ignoring the best and worst performance, and calculating the average of the rest. Each experiment involves retraining the model from scratch, not just the inference.

The training was performed from scratch on the full NeuMF model without the use of pretraining. While pretraining helped improved the performance slightly on the original dataset, it requires a lot of time to re-train both the GMF and the MLP before training the NeuMF, and therefore we chose not to utilize it. With pertaining, we would expect a small increase on the performance of all experiments.

VI. Performance Comparison

A. Experiments with variable top-k

By examining figure 3 along with its corresponding table II, we observe the same impact of the top-k's value on the performance as that obtained from the original experiment with the MovieLens dataset. Specifically, the more potentially interesting items we can recommend to a user, the more likely it becomes to provide at least a few accurate recommendations. Nonetheless, the performance increase rate gradually diminishes for larger values of k, thus showing a tendency of the performance to eventually stabilize.

The only difference is that in the original experiment, HR@K ranges from approximately 0.2 to a maximum value of over 0.7, and NDCG@K ranges from about 0.2 to a maximum value of around 0.45. This slight decline in performance when using that subset of MovieLens is expected due to the smaller ratings per user ratio that was discussed previously.

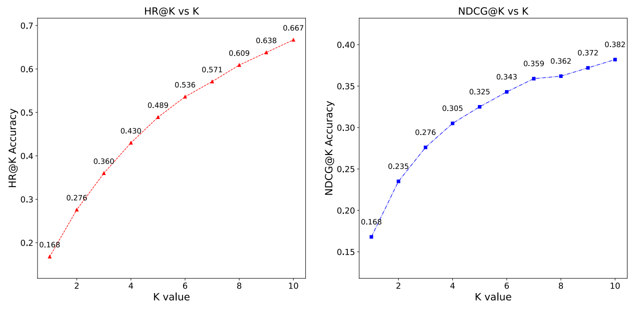

Fig. 3. Left: y-axis HR@K (the higher the better), x-axis K. Right: y-axis NDCG@K (the higher the better), x-axis K. In both plots K takes values from 1 to 10 with stride equal to 1.

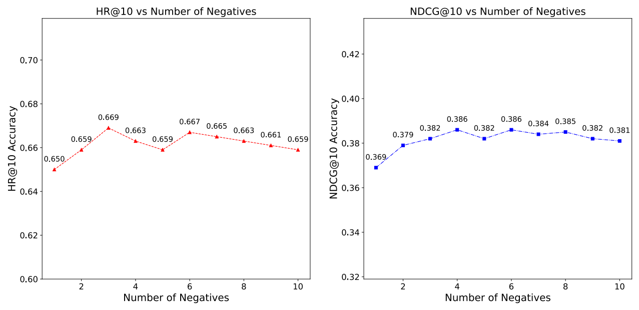

Fig. 4. Left: y-axis HR@10 (the higher the better), x-axis number of negatives. Right: y-axis NDCG@10 (the higher the better), x-axis number of negatives. In both plots, the number of negatives takes values from 1 to 10 with a stride equal to 1.

B. Experiments with variable number of negatives

Regarding the influence of negative sampling on the accuracy, it is apparent in figure 4 that a slight increase in the negative samples can yield respectable accuracy improvements. However, an aggressive attempt to do so causes fluctuations in accuracy and can quickly lead to overfitting. Additionally, it is worth mentioning that inserting more negative samples slows down considerably the training process, thus making it preferable to keep that number small in order to boost performance without paying a crucial time penalty in training. In our case, we should not exceed 4 negative samples to achieve those qualities.

While similar conclusions were reached by the same experiment for the original dataset, it once again led to a slightly more significant performance boost and had relatively less sharp fluctuations. Again, we attribute that faint decline in accuracy to the smaller mean ratings per user ratio, which translates to less information on average about a user's profile.

References

[1] MovieLens Dataset

[2] He, X., Liao, L., Zhang, H., Nie, L., Hu, X. & Chua, T. Neural Collaborative Filtering. *Proceedings Of The 26th International Conference On World Wide Web*. pp. 173-182 (2017), https://doi.org/10.1145/3038912.3052569