RecNet: A Deep Learning Based Collaborative Filtering Recommender System

University of the Witwatersrand
School of Computer Science and Applied Mathematics

B.Budler 1827655, T. Packirisamy 1839434

July 16, 2021

Abstract

As the amount of information available to online users has increased significantly, systems which recommend the most appropriate items to a user have become imperative. One approach to making these recommendations is using similar users' preferences, known as Collaborative Filtering. Recently, Deep Learning has shown promising results in improving Collaborative Filtering Recommender Systems. In this paper, a novel Deep Learning based Collaborative Filtering Recommender System, RecNet, is proposed and compared to a traditional Collaborative Filtering approach using the MovieLens100k dataset.

1 Introduction

With an increasing amount of online data, Internet users are faced with an information overload when deciding what content to engage with [Mu 2018]. A Recommender System (RS) is a vital information filtering tool which helps ameliorate this problem by providing a personalized user experience [Zhang *et al.* 2019]. Deep Learning (DL), an emerging field of Machine Learning, has become the state-of-the-art technique in several different domains including natural language processing, speech recognition, image processing and object recognition [Batmaz *et al.* 2019]. After an initially slow uptake by the RS research community, DL has recently become a widely popular approach in the RS domain [Karatzoglou and Hidasi 2017]. This is largely due to the fact that multi-layer DL models are able to extract abstract feature representations of both user generated data and item data to provide better recommendations than traditional RS approaches [Karatzoglou and Hidasi 2017].

Recommender Systems are typically divided into two categories; Content-based and Collaborative Filtering (CF). Content-based systems rely on the attributes of items and recommend items with similar features to other items a user has interacted with [Mu 2018]. CF, on the other hand, relies on the information and behaviour of other users. Recommender Systems built using CF determine which users are similar to the active user and then make recommendations to the active user using items that have been positively rated by similar users [Batmaz *et al.* 2019]. There are also hybrid approaches which incorporate aspects of both content-based and CF.

In this research RecNet is presented, a Deep Learning-based Collaborative Filtering Recommender System which utilizes embeddings of explicit user feedback and a Deep Neural Network to make recommendations. A comparative analysis between the proposed model and a traditional CF approach on a benchmark RS dataset, the MovieLens 100K, is also included to further investigate the improvements that Deep Learning can make in the Recommender System domain.

The rest of this report is structured as follows; Section 2 explores academic literature related to the application of Deep Learning for Collaborative Filtering in Recommender Systems. Section 3 outlines the data to be used in the research as well as the methodology that was followed to develop both the traditional Collaborative Filtering system and the proposed RecNet model. Section 4 contains an evaluation of the RecNet model and a comparison between the two systems. Lastly, Section 5 concludes with a brief overview and closing remarks.

2 Related Work

The pivotal work done by Cheng *et al.* [2016] served as a catalyst for research into the applicability of Deep Learning for Recommender Systems. The RS they propose was developed for the Google Play mobile app store and combines the memorization benefits of a wide linear network with the generalizability of a Deep Neural Network, a process coined as Wide and Deep Learning.

Guo *et al.* [2017] built on the foundations of the Wide and Deep Learning model and used shared inputs for both the wide and deep aspects of the model, allowing the costly feature engineering to be reduced. Lian *et al.* [2018] expanded on this idea and developed a model capable of learning both explicit and implicit high-order feature interactions, further reducing the feature engineering required and outperforming both of the previous implementations.

He *et al.* [2017] argue that the previous implementations can all be represented in a general framework where the sparse implicit user and item representations are first projected into dense embeddings before feeding these embeddings into a multi-layer neural architecture which they call *Neural Collaborative Filtering*. The authors demonstrate how the previous approaches can be modeled in this way and outline that their framework opens a new avenue for future research into DL-based Recommender Systems.

The RecNet model proposed in this research utilizes the Neural Collaborative Filtering framework but incorporates the explicit ratings given by users as opposed to the implicit representations. This alteration is made on the foundation that personal privacy issues are gaining more attention from Internet users [Mu 2018]. Tracking the indicators which contribute to implicit feeedback, such as when a user clicks on an item or watches a video, may become prohibited by privacy laws.

3 Methodology

3.1 Data

The MovieLens datasets have become the most popular datasets for evaluating Recommender Systems [Zhang *et al.* 2019]. There are four datasets available namely the MovieLens100K, MovieLens1M, MovieLens10M and MoviesLens20M where the suffix indicates the number of discrete ratings each dataset contains.

In this research the MovieLens100k (ML100K) dataset is used to evaluate both the traditional CF approach as well as the proposed RecNet model as it is sufficiently sparse but allows for a reduction in computational complexity. The ML100K consists of 100 000 ratings given to 9742 movies by 610 users. Each record in the dataset consists of a User ID, a Movie ID, a rating between 0.5-5 in increments of 0.5 and a timestamp at which the rating was given. As neither of the systems in this research incorporate the temporal aspect of the data, the timestamp feature is dropped.

3.2 Recommender Systems

The general framework in which CF Recommender Systems are studied involves a user-item matrix. Here, a user-movie matrix is used to for ease of explanation. The columns in the matrix represent the set of users $U = \{u_1, u_2, ..., u_i\}$ in the dataset while the rows represents the set of movies $M = \{m_1, m_2, ..., m_j\}$. The value of the position r_{ij} in the matrix corresponds to the rating a user u_i gave to the movie m_j. This matrix is typically sparse, as a single user has only rated a limited number of movies. The objective of the RS is to predict the rating a user would give to a movie they have not rated yet and use the highest predicted ratings to make recommendations.

	u_1	u_2	u_3	...	u_i
m_1	4		3	2	
m_2		1			5
m_3			2		4
...	3			1	
m_j		2	5		r_{ij}

Table 1: User-movie Matrix

3.3 Collaborative Filtering

The development of a CF Recommender System typically includes two phases. The first phase involves quantifying the similarity between users while the second phase utilizes this similarity to predict ratings and make recommendations using these predicted ratings.

3.3.1 Quantifying User Similarity

One approach to determining the similarity of users is to represent each user as a j dimensional vector where j represent the number of movies in the dataset. The similarity between two users can then be calculated using the angle between two of these user vectors [Zhang *et al.* 2008]. The smaller the angle, the more similar the users. The angle between two user vectors u_1 and u_2 can be calculated using the cosine as follows:

$$cos(u_1, u_2) = \frac{u_1 \bullet u_2}{||u_1|| \cdot ||u_2||}$$

where $\bullet$ represents the dot product between the two vectors and $||u_1||$ is the l2 norm, or length, of vector u_1. This approach is known as the cosine similarity [Schwarz *et al.* 2017].

A major issue when using the cosine similarity approach is that some users will, in general, rate movies higher than average while some users will be more harsh and rate movies lower than average. This disparity leads to a bias in the results obtained using the cosine similarity metric. To account for this bias, each user's ratings need to be normalized. This can be done by subtracting the mean rating of a particular user $\bar{u_i}$ from each of their ratings. When these adjusted ratings are used to form the user rating vectors, the cosine similarity between two vectors is known as the centered cosine and can be represented as:

$$cos(u_1, u_2) = \frac{(u_1 - \bar{u_1}) \bullet (u_2 - \bar{u_2})}{||(u_1 - \bar{u_1}))|| \cdot ||(u_2 - \bar{u_2})||}$$

This centered cosine formula is the same as that of the Pearson Correlation Coefficient and empirical studies have demonstrated that it is an effective way of determining user similarity [Breese *et al.* 2013].

3.3.2 Predicting Ratings

Once a subset of users similar to the active user has been selected the rating, r_{aj}, that the active user, u_a, would give a movie they have not yet rated, m_j, can be predicted by calculating the mean of the ratings given by the k most similar users:

$$r_{aj} = (\sum_{i=0}^{k} r_{ij})/k$$

However in most situations, some of the users will be significantly more similar to the active user than others. Simply finding the mean of the most similar ratings is not be an appropriate choice for predictions. To compensate for this difference in similarities, the rating prediction calculation can incorporate the similarity measure. This is done by weighting a similar user's rating according to how similar they are to the active user and using this weighted sum of ratings instead. Let S_i denote the similarity between a user u_i and the active user u_a, the weighted sum approach for predicting ratings is then given as:

$$r_{aj} = (\sum_{i=0}^{k} r_{ij} \cdot S_i)/(\sum_{i=0}^{k} S_i)$$

.

The traditional CF approach that is used as a baseline in this research uses the Pearson Correlation Coefficient to determine the most similar users and then incorporates a weighted sum of the 5 most similar users' ratings to predict ratings for the active user.

3.4 Neural Collaborative Filtering

Similar to traditional Collaborative Filtering (CF), Neural Collaborative filtering (NCF) [He *et al.* 2017] predicts and recommends items based on similar users' preferences. The proposed NCF method makes use of a Multi-Layer Perceptron (MLP) to combine latent features of user and item data in a way that optimizes predicting a correct rating for a given user and item pair. The latent features are calculated using an embedding layer.

3.4.1 Embeddings

User and item values can be seen as categorical variables which need to be represented as numerical values. The traditional method used to represent a categorical variable numerically is to form a vector with the dimension equal to the number of categories. The vector contains zeros in all positions aside from the index corresponding to the given category. This method is known as one-hot encoding [Potdar *et al.* 2017]. This method becomes inefficient in the case of movie recommender systems where the number of users and movies are characteristically high. The fact that there are a great number of categories means that each representation of a category will be incredibly sparse, making learning more difficult and increasing the computational cost of the system. It is therefore required to extract latent features from each category to create a numeric representation.

Embeddings provide a method of handling categorical variables that is suitable to the characteristics of Recommender Systems. Embeddings convert each category of a categorical variable into a dense N-dimensional vector of latent features. This vector represents the category in the latent space and distances between categories can be calculated. This, therefore, means that the network can learn and update embeddings in order to place similar users closer together within the latent space. The same process can be applied to the movies.

3.4.2 Multi-Layer Perceptron (MLP)

Collaborative Filtering systems use a variety of different methods to make predictions using the combined knowledge about a user and a movie but NCF employs a Multi-Layer Perceptron (MLP). The MLP section of the network consists of fully connected layers which learn to map the relationship between a user's embeddings and a movie's embeddings to a predicted rating for the pair. This method is superior, in terms of performance, to the traditional Collaborative Filtering approach as well as the common Matrix Factorisation method [Koren *et al.* 2009] which employs a fixed inner product to compute predicted ratings. During training, the network learns by backpropogating the observed residuals through the MLP layers and the embedding layers, to move similar users and items closer together and reduce the error.

4 Evaluation

4.1 Performance Metric

The Root Mean Squared Error is used as the performance metric throughout this evaluation and is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(r_i - \hat{r_i}^2)}{N}}$$

where $\hat{r_i}$ is the rating predicted by the model, r_i is the actual rating given by a user to a specific movie and N is the number of ratings in the dataset.

4.2 Initial NCF model

The first implementation of the NCF model made use of 32 embeddings which were fed into an architecture of 2 hidden layers of 32 units and an output regression unit. A ReLU activation function was employed. Figure 1 illustrates how the prediction error decreases as the network learns. It can also be seen that after around 25 batches the network begins to outperform the baseline traditional CF system.

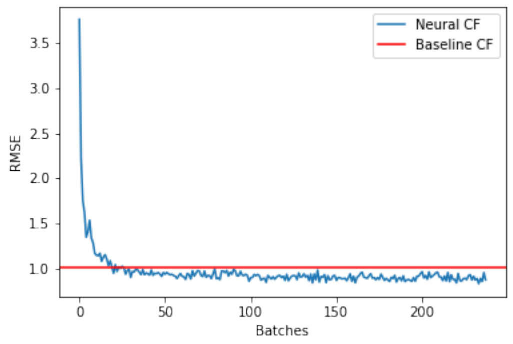

Figure 1: Inital NCF Training Loss

4.3 NCF Model Improvements

This section describes the improvements made to the network based on parameter and architectural experimentation. Each test is performed using a cross-validation set to ensure the model does not simply optimize for the test set.

4.3.1 Model Architecture Comparison

A variety of different architectures were tested and their optimization over 80 batches can be seen in Figure 2. It is evident that the best performing architecture is the model containing layers of 64, 32 and 16 units that lead into a single output unit.

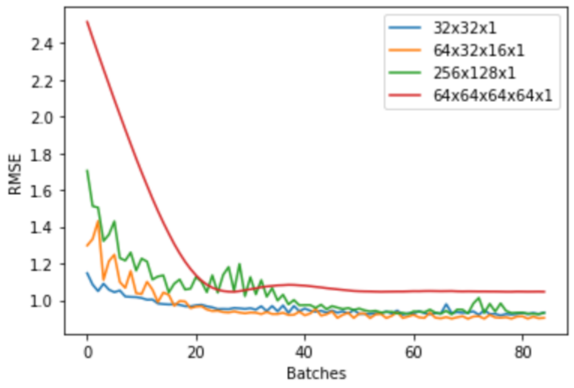

Figure 2: Cross-validation Loss for Different NCF architectures

4.3.2 Tuning Batch Sizes

The model was trained on a variety of batch sizes and the final losses after training were compared. This comparison can be found in Figure 3. It can be seen that a batch size of 1024 performs best during optimization as it achieves the lowest final error on the cross-validation set.

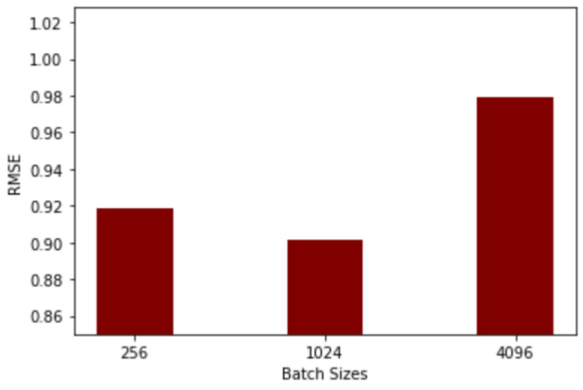

Figure 3: Cross-validation Losses for Different Batch Sizes

4.3.3 Tuning Learning Rate

Figure 4 indicates the losses attained using different learning rates in the construction of the model. Evidently, a learning rate of 0.1 results in the fastest convergence.

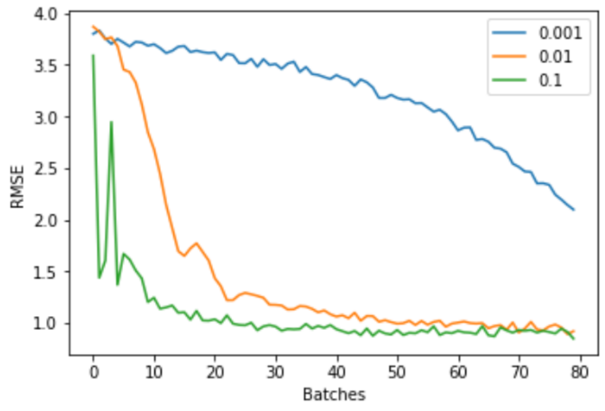

Figure 4: Cross-validation Losses for Different Learning Rates

4.3.4 Regularisation

Using the architecture, batch size and learning rate above, the model was evaluated using the test data. A relatively large difference between the train and test data was observed which is indicative of over-fitting. To alleviate this over-fitting, various regularisation techniques were employed.

The first of these regularisation techniques was L2 regularisation. As can be observed in Fig 5, this technique was able to significantly reduce the gap between the train and test error but overcompensated resulting in under-fitting and a higher RMSE.

The second regularisation technique, however, involved the incorporation of dropout layers and was able to slightly reduce over-fitting, hereby increasing generalizability, without sacrificing on accuracy.

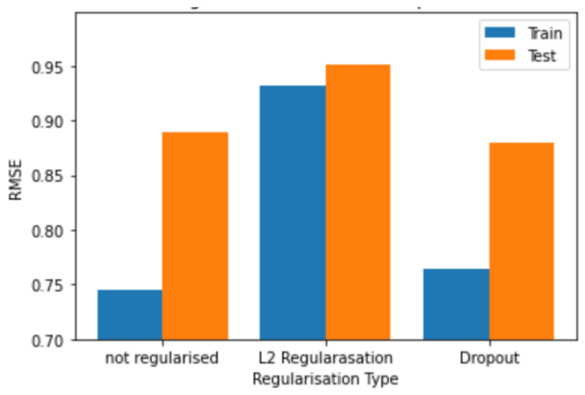

Figure 5: Train and Train Losses for Different Regularisation Techniques

4.4 Final NCF Model - RecNet

Based on the experiments in the previous section, the final model was composed. The model has an architecture that consists of an embedding layer with 64 units (32 for user embeddings and 32 for movie embeddings). This is followed by fully connected layers of 64, 32 and 16 units respectively which lead into a single linear unit that outputs the predicted rating. A ReLU activation function is utilized between layers and a dropout factor is incorporated to prevent over-fitting. The batch size of 1024 is used along with a learning rate of 0.1 to train the network using Adam optimisation [Kingma and Ba 2017].

Figure 6 shows the prediction error of the final model decreasing while the network learns. It can be seen that, by the end of training, RecNet significantly improves on the performance of the traditional baseline Collaborative Filtering algorithm.

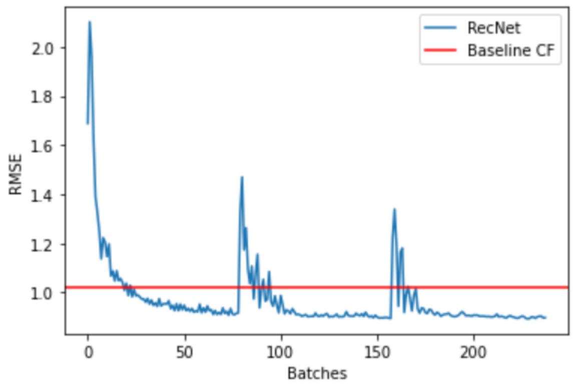

Figure 6: Final NCF (RecNet) Training Loss

Table 4.4 indicates that the various improvements made to the NCF model have successfully decreased the RMSE on the test set from 0.9396 to 0.8976. Both implementations offer significant improvements over the baseline.

Model	Train RMSE	Test RMSE
Baseline CF	-	1.0204
Initial NCF	0.8008	0.9396
Final NCF (RecNet)	0.7674	0.8976

5 Conclusion

As can be seen from the empirical evaluation above NCF was able to achieve significantly better results than the traditional Collaborative Filtering approach. These results were further enhanced through the optimization of network architectures and hyperparameters as well as the incorporation of regularisation techniques. RecNet demonstrates the improvements that Deep Learning can make in the Recommender System domain and serves as a baseline for further NCF research. Rec

References

[Asanov and others 2011] Daniar Asanov et al. Algorithms and methods in recommender systems. *Berlin Institute of Technology, Berlin, Germany*, 2011.

[Batmaz *et al.* 2019] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, 52(1):1–37, 2019.

[Breese *et al.* 2013] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.

[Cheng *et al.* 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[Guo *et al.* 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[He *et al.* 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. *Neural Collaborative Filtering*, 2017.

[Karatzoglou and Hidasi 2017] Alexandros Karatzoglou and Balázs Hidasi. Deep learning for recommender systems. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 396–397, 2017.

[Kingma and Ba 2017] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*, 2017.

[Koren *et al.* 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[Lian *et al.* 2018] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1754–1763, 2018.

[Linden *et al.* 2003] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[Mu 2018] Ruihui Mu. A survey of recommender systems based on deep learning. *Ieee Access*, 6:69009–69022, 2018.

[Potdar *et al.* 2017] Kedar Potdar, Taher Pardawala, and Chinmay Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, 175:7–9, 10 2017.

[Schwarz *et al.* 2017] Mykhaylo Schwarz, Mykhaylo Lobur, and Yuriy Stekh. Analysis of the effectiveness of similarity measures for recommender systems. In *2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, pages 275–277. IEEE, 2017.

[Zhang *et al.* 2008] Liang Zhang, Bo Xiao, Jun Guo, and Chen Zhu. A scalable collaborative filtering algorithm based on localized preference. In *2008 International Conference on Machine Learning and Cybernetics*, volume 1, pages 160–167. IEEE, 2008.

[Zhang *et al.* 2019] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.