Exploring the Performance of Deep Learning Recommender Systems for Film Recommendations

Gianolli Dorcelus
Dept. of Computer and Software Engineering
Polytechnique Montréal
Montréal, Canada
gianolli.dorcelus@polymtl.ca

Mael Cossec
Dept. of Computer and Software Engineering
Polytechnique Montréal
Montréal, Canada
mael.cossec@polymtl.ca

***Abstract*—Recommender systems are known to be widely used in most of today's systems. This study aims to explore the performance of such systems by conducting a replication of two approaches that use deep learning to recommend videos. Our goal was not to produce something new, but to be able to understand and adapt the two models to our dataset, and to have results that are, at least, coherent. As results, while Collaborative Deep learning models are more prone to recommend videos related to the preference of the users, Candidate_Generation model in Deep Neural Networks provides recommendation based on popularity.**

***Index Terms*—Recommender systems, replication, Artificial Intelligence,**

I. Introduction

Recommendations of contents have seen an increasing adoption and popularity those recent years on many web sites and systems. Social media benefit from tailored recommendations to bring users closer, to build network, to share relevant advertisements... Companies adopt recommendation algorithms to improve e-commerce metrics [1]. Moreover, million of users use streaming service actively on a daily basis resulting in a huge amount of information to be analyzed. Since Artificial Intelligence techniques have proved efficient to deal with large-scale data, it becomes crucial to adopt these approaches to provide personalized and relevant content to users.

By the past, matrix factorization was the baseline for recommender system where recommendations were only related to what a particular user sees and rates. Algorithms of nowadays also focus on customer features (age, sex, demography, ...) to bring much suitable recommendations.

This study aimed to analyze the performance of deep learning recommender systems that involve in recommending videos. Hence, we conducted a replication of two different papers:

1) Covington et al. [2] provided a combination of two neural networks to recommend Youtube videos. The first neural network named *"Candidate Generation"*, predicts hundred relevant videos for a user based on large corpus videos (millions) and the user activity history. The second Neural Network *"Ranking"* is fed by the *"Candidate Generation"* and the user activity history to assign a score to the candidate videos and recommend those with the highest score.
2) Wang et al. [3] were the first to combine deep learning with collaborative filtering. Collaborative filtering uses the ratings of the items by the users, to produce recommendations. The deep learning part is a Stacked Denoising AutoEncoder (SDAE), that uses information about the items to improve the recommendations. In fine, the model combines the two approaches to predict the whole rating matrix and therefore be able to produce recommendations.

This paper is organised as follows. The section II presents other studies performed around the same area. The section III describes the methodology of this paper to achieve the purpose. The section IV highlights the limitations of the paper. And finally section V and section VI present respectively the results and a brief conclusion of the work.

II. Related Work

A lot of recent papers mention the use of Deep Neural Network in Collaborative Filtering or for Recommender Systems in general. He et al. [4] proposed a framework called Neural network-based Collaborative Filtering, which replaces the traditional inner product with a neural architecture capable of learning complex functions directly from data by capturing interactions between user and item features without using matrix factorization and its weaknesses. While Deep Neural Networks can automatically learn feature interactions, they generate these interactions implicitly and may not efficiently capture all types of cross features. Wang et al. [5] introduced Deep and Cross Network (DCN) model, which kept the advantages of DNNs and incorporated a novel cross network (apply a feature crossing at each layer) to avoid the need for manual feature engineering.

Yuhan et al. in this paper [6] introduce a genetic algorithm to design Deep Neural Networks in the context of Collaborative Deep Learning. Zhang et al. [7] provide a good review of the research on deep learning recommender systems as of 2018.

Traditional collaborative filtering methods typically focus on learning embeddings through pairwise interaction factorization, Liu et al. [8] introduced a recommendation method called Collaborative Filtering with Dual Autoencoder (CFDA) that used dual autoencoders to simultaneously learn hidden

representations of users and items and tried to capture the complex relationships between users and items.

All in all, deep learning is now really present in the world of recommender systems and is a really important aspect to take into account, in order to create good and efficient recommendations.

III. Methodology

In this section we will describe how we replicated the steps taken to collect the data and to do the performance analysis of the 2 systems. We will also explain where we deviated from the original papers and why in terms of methodology.

A. Data Collection

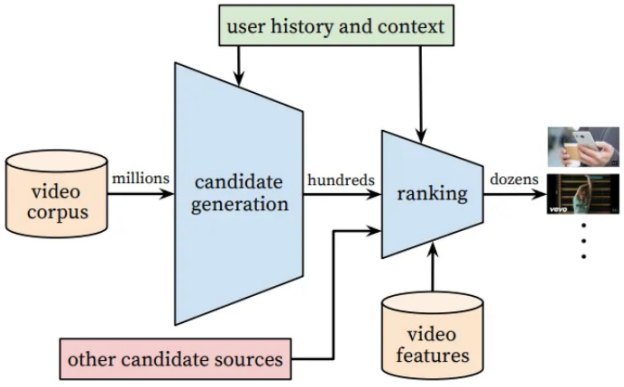
Opposite of both replicated papers, we extracted data from Kaggle [9] that is an online platform for data scientist community. *Full MovieLens* dataset [10] consists of 26 million ratings (scale of 1-5) and 750,000 tag applications from 270,000 users on 45,000 movies. *Full MovieLens* data was collected from TMDB and GroupLens from 2017 and earlier. The files are available in csv format:

- movies_metadata.csv: The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include genres, release date, ...
- ratings_small.csv: The subset of 100,000 ratings from 700 users on 9,000 movies.
- ratings.csv: The whole ratings

B. Deep Neural Networks Approach

The authors of the first original paper [2] combined two neural networks with the purpose of providing recommendations.

Fig. 1. YouTube Recommendation System Architecture



1) Data pre-processing: To feed the Candidate Generation network, in accordance to the original paper, we created a limited sequence of video ids for each user that are related to the user's preference. To achieve this, since in the MovieLens ratings file, each row corresponded to a rating of a movie by a user, we converted it in a new file with a row corresponding to a single user with a list of movies that the user has watched, a list of movie titles, a list of ratings associated to the movies, and a list of movie genres.

However, contrary to the original paper, and since these features weren't available in our dataset, we didn't add age, occupation and demographic data to the input. But future work can attempt to analyze the significance of different features and concatenate them with other inputs for better learning.

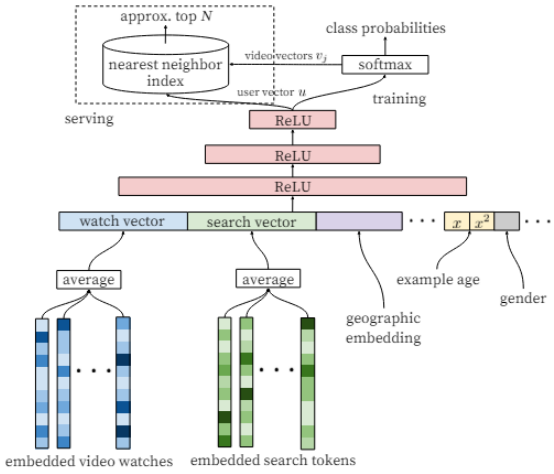
Hence, we transformed the movie ratings into binary classes, given a rating greater or equal to 3 were considered as a liked video, and disliked otherwise. The list of videos for each user, then was divided into like and dislike videos. A new column is added to the data *predict_label* that contained the last like video each user has watched.

Fig. 2. An input sample for Candidate Generation Network

	user	dislike	like	title_d	predict_labels
0	0	[0, 1, 2, 4, 5]	[]	[0, 1, 2, 3, 4, 5]	3
1	1	[24, 42, 56, 63]	[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1...	[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1...	62
2	2	[37, 80]	[64, 65, 66, 67, 68, 69, 70, 7, 40, 71, 72, 73...	[5, 62, 63, 64, 65, 66, 67, 7, 39, 68, 69, 70,...	81
3	3	[118, 119, 137, 145, 168]	[82, 83, 84, 85, 86, 87, 88, 89, 90, 2, 8, 91,...	[5, 78, 79, 80, 81, 82, 83, 84, 85, 2, 8, 86, ...	181

2) Model Architecture: With the exception of data omitted for input concatenation, we followed the similar architecture

Fig. 3. An input sample for Candidate Generation Network



1) Input layer: Where the encoded video sequences are received. To prepare the data for the neural network, sequences are padded to the same length within each batch. This uniform length allows for consistent input size into the network.
2) Embedding layer: In this layer, the input data is transformed into a lower-dimensional space based on the specified dimensions in the layer configuration. This transformation helps in reducing the complexity of the data while retaining important features.
3) L2 normalisation layer: This layer is used for regularization, aiming to keep the weights of the network smaller. This ensures that the network learns primarily from the actual data, rather than from the padded values.

4) Embedding aggregator: That aggregated the embeddings by taking the average of the embeddings component-wise. According to the original paper, this averaging method was found to be the most effective way of combining the embeddings.
5) Concatenate layer: Combined the outputs from all the averaged embedding layers and other signals to create a unified input for the subsequent layers in the network.
6) ReLU (Rectified Linear Unit) Layer: To add non-linearity into the network, allowing it to learn complex relationships in the data.
7) Batch Normalization Layer: This layer normalized the input to each subsequent layer. It helps in stabilizing and accelerating the training process by ensuring that the distribution of input values remains consistent across batches, to prevent significant variations in the shape of the distribution.
8) Dense output layer: Generated predictions in the form of probabilities across all available movie classes. It utilized a softmax activation function to produce these probabilities, allowing the model to assign likelihood scores to each class.

Finally, we used sparse_categorical_crossentropy to try to reduce the loss between the next predicted video and the actual video watched by the user from the video corpus.

3) Experiments: The experimental phase of this study was performed using TensorFlow (Version 2.15) with a computer having the following characteristics: RAM 8.00 GB, Intel(R) Pentium(R) Gold 7505 @ 2.00GHz. In this experimental phase, we used an implementation available on GitHub by Hyez. Since our dataset didn't have features related to the users, instead of training two distinct neural networks to make recommendations. We only train a single neural network by feeding it with like and dislike videos for each particular user. The complete files are available in the package [11] in order to guide future studies. Figure 4 presents the Candidate Generator Network Architecture :

4) Evaluation: The network mentioned above were evaluated firstly by training 80% and testing 20% of the total data. And for the sake of exploring the performance of the system, we evaluated it considering the similarity between the genres of recommended videos and the genres of videos already liked by the users.

C. Collaborative Deep Learning approach

The authors of the original paper [3], combined a collaborative filtering approach with a content based approach using deep learning, to produce recommendations. They believed that the two approaches were complementary and that combining the two would improve the predictions of the ratings.

1) Data pre-processing: To feed the CDL system, we need two matrices, the first one is the ratings matrix and the second one is the film-content matrix. To simplify the use of the model, we decided to only keep the genre of the movies in the content matrix, i.e. Drama, Comedy, Action, etc...

Fig. 4. Candidate Generator Network Architecture

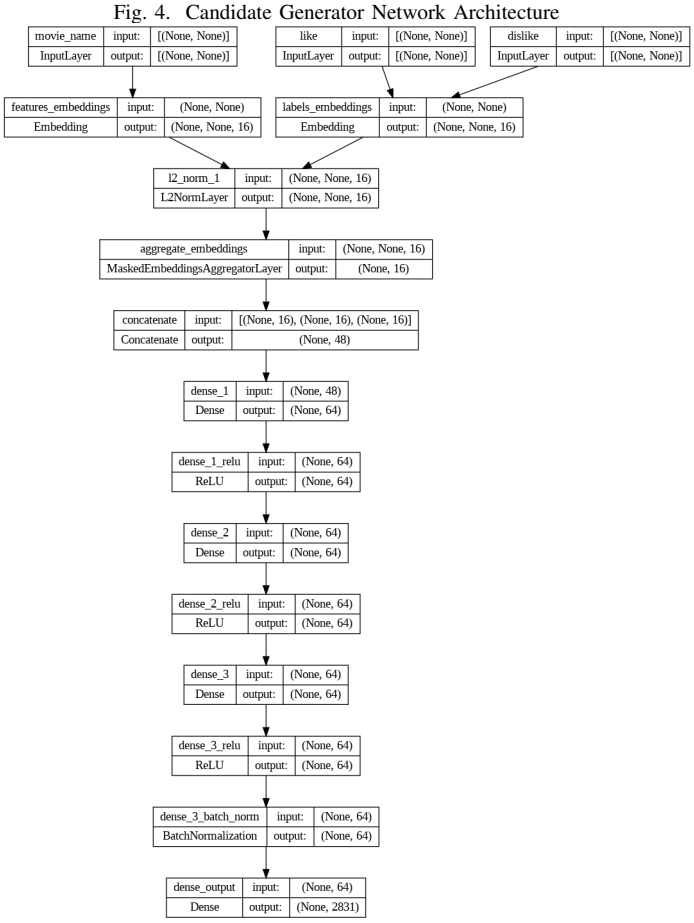
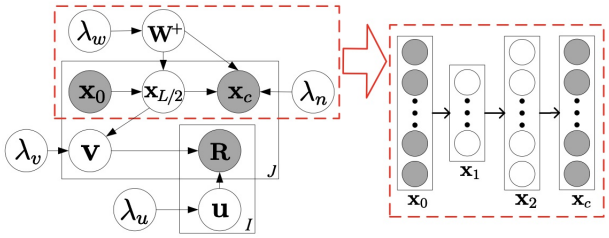


Fig. 5. Collaborative Deep Learning Architecture with the SDAE inside the dashed rectangle and R the rating matrix.



We first had to create a matrix using the movies_metadata.csv file and filter everything that wasn't interesting, and to format the data so that we have a clean, film-content matrix containing film ids and their genres. Then we filtered the films to have only those that already had ratings in the ratings_small.csv file but that also had information about them in the film-content matrix. We kept the films that had both.

After that we made one column by genre of films existing, so that the film-content matrix was a binary matrix, in which each row was corresponding to a film and it only had ones in

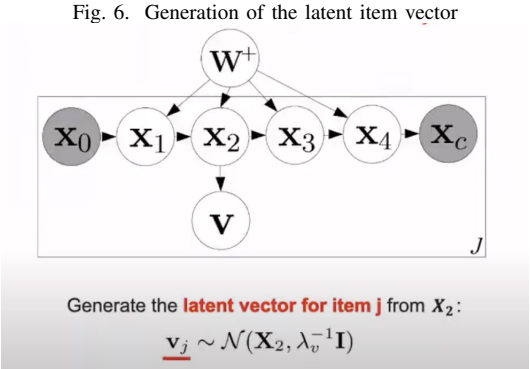
the columns of the movie's genres.

Finally we had the rating matrix and the film-content matrix, with the same corresponding movies in them, to feed to the model.

2) Model Architecture: We have two different part in this architecture, one with the SDAE which represent the content approach and one that represent the collaborative filtering.

1) SDAE: it is a feedforward neural network composed of one encoder and one decoder. It learns the feature representation, but is also able to capture similarities and relationships between the items. We add noises to the film-content matrix and we then feed it to the SDAE, to recreate the original matrix. However here, we are interested in the middle layer representation.

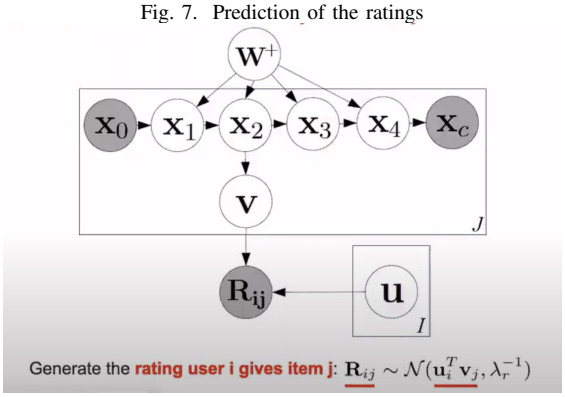
Fig. 6. Generation of the latent item vector



The weights are updated using Gradient Descent in the implementation that we found. The latent item vector created will be used in the collaborative filtering part in combination with the user vector, to predict ratings.

2) Collaborative filtering: now that we have the latent item vector, we can use it in combination with the user vector to generate our rating matrix and predict the ratings. We basically do the product of the two vectors to generate the rating of one item by one user.

Fig. 7. Prediction of the ratings



Now we have predicted the ratings of all items by all users and we are now able to make recommendations for each users.

3) Experiments: To experiment with CDL, we used an implementation found on GitHub by Chen Bo Syun [12], we had to adapt our dataset to fit this implementation, as described before. It uses TensorFlow (Version 1) and we used a computer with the following characteristics: RAM 8.00 GB, Apple M1 chip. We first built the model, then trained it.

4) Evaluation: After we made our predictions, we sort the predicted ratings for each user and we recommend the top M items. We then considered the recall@M defined as :

$$recall@M = \frac{number\ of\ items\ the\ user\ likes\ in\ the\ top\ M}{total\ number\ of\ items\ the\ user\ likes}$$

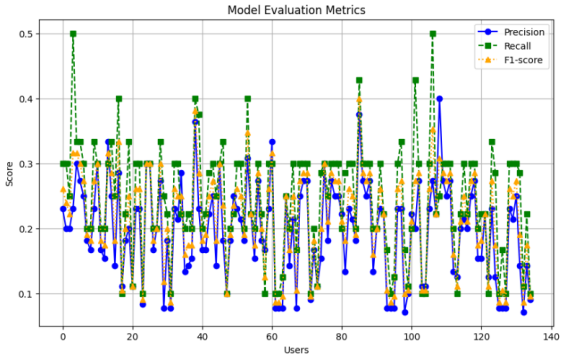
IV. Results

We successfully managed to have results for both approaches, using the implementations that we found on GitHub.

A. Deep Neural Networks

This approach, was used by the past to recommend YouTube videos and our results agreed with this since the recommended videos provided for each user reflected the at some point the preferences of the user. Moreover, the model are also prone to bring out new and popular videos to that particular user. Figure 8 presents the results and shows better recall than precision, that fits our specific environment since it helps assessing how effective the model is at capturing relevant videos among the user's preferences and interests.

Fig. 8. Model Evaluation Results



B. Collaborative Deep Learning

For this method, we managed to plot the average recall@M and the average precision@M over all users:

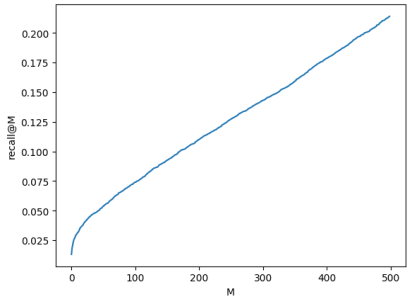


Fig. 9. Average Recall

We have a correct recall, that is not too low, taking into account the context of our predictions.

V. Threats to Validity

This section highlights the threats to validity that can possibly bring bias to our study. In the case of Candidate Generation, the original paper considered user's features(age, demography, occupation, search historic ..) to provide recommendations and then utilized video's features (genre, rating, timestamp,...) to rank the recommendations. Since our dataset didn't have any information related to the users. We first, train the model by feeding it with some video features, however the preliminary results shown instability in the training phase. Hence, to mitigate the threat we decided to used a single neural network. All the stages of this work are detailed in the section III.

VI. Conclusion

In this paper, we managed to implement and adapt two Deep Learning methods for recommender systems. It's pretty hard to evaluate the results obtained as we don't have other references for our specific context, but we think we managed to have correct results in fine. Moreover, since the dataset didn't perfectly match the available implementations, our results didn't perfectly reflect those presented in the original papers. However, overall this study can serve as baseline for future improvements or studies.

References

[1] https://theiconic.tech/implementing-the-youtube-recommendations-paper-, [Online; accessed 19-April-2024].

[2] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.

[3] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1235–1244.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[5] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proceedings of the ADKDD'17*, 2017, pp. 1–7.

[6] Y. S. Yuhan Fang, Yuqiao Liu, "Evolving deep neural networks for collaborative filtering," in *In Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part VI 28*, 2021, pp. 230–238.

[7] A. S. Shuai Zhang, Lina Yao and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," in *ACM Comput. Surv. 1, 1, Article 1 (July 2018), 35 pages*, 2018.

[8] X. Liu and Z. Wang, "Cfda: collaborative filtering with dual autoencoder for recommender system," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–7.

[9] https://www.kaggle.com/, [Online; accessed 19-April-2024].

[10] http:https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset, [Online; accessed 19-April-2024].

[11] https://github.com/Cussex/Projet-INF8225.

[12] https://github.com/ChenBoSyun/implement-Collaborative-Deep-Learning-, [Online; accessed 19-April-2024].