Recommendation System using Deep Learning

Ammar Haziq Bin Mohd Halim
Student of Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
ammar-haziq.bin-mohd@stud.hshl.de

***Abstract*—Recommendation systems have become an integral part of ones daily life. This is because it solves the problem of information overload and helps consumers in the decision process. With the help of deep learning, one can design the system with improved performance relative to classical methods. This seminar paper aims to give the reader an understanding of recommendation systems and also provides a basic intuition of how deep learning models could be applied to design said systems. The paper starts with a formal introduction to recommendation systems and deep learning as preliminaries so that the reader is well equipped to understanding the implementation part of the paper. Then the paper goes into detail of one implementation that was done as to paint a picture on the whole process of using deep learning for recommendation systems**

***Index Terms*—Recommendation systems, Deep Learning**

I. Introduction

Information overload continues to be an issue in the era of big data. With so many options available to Internet users on a daily basis, it may be difficult for them to pick and choose relevant information, impeding their access to items of interest on the Internet. This has led to the development of recommendation systems to aid in the resolution of this issue. Recommendation systems are information filtering systems that provide the user with a list of recommended items. A recommendation system can exist as an artificial intelligence or AI programme, typically connected with machine learning, that leverages Big Data to recommend or suggest additional items to users.

Recommendation systems have diverse applications and are essential for businesses to promote things that are informative and appealing to potential customers. Use cases for recommendation systems include the e-commerce, entertainment, and social media industries. Today, numerous large technology companies have their own recommendation systems. Multiple cases include Netflix, which uses the system to propose intriguing movies to its viewers, Spotify, which makes customized playlists for its users, and Amazon, which suggests products to be purchased. This has assisted companies in increasing their profit margins.

This paper is structured as follows, the next section the paper introduces the reader to the overview of recommendation systems to allow the user to get to know the different types of recommendation systems that exists.Section 3 consists of some core fundamental knowledge on deep learning. This is to give the user an understanding of how neural networks learn. Section 4 dives in the implementation of a movie recommendation system to give an insight on how to apply the theory of deep learning in the specific domain. Then in section 5, an analysis of the results of training using evaluation metrics is given. Finally, The last section is the conclusion of this paper

II. Overview of Recommendation Systems

Like many machine learning problems, in order to design a good recommendation system, one must have access to not only large, but also good quality data for the system to learn. For recommendation systems, the data that is useful is based on user feedback. This is crucial in the development of said systems as it relies on the user preferences to give meaningful suggestions to the user. These may be based on past purchases, search history, demographic data, and other variables. To interpret user preferences, there exists two types of data that are important for recommendation systems in the information collection phase which are Explicit feedback and Implicit feedback.

Explicit feedback is feedback that relies on the user ratings. Typically, the system invites the user through the interface to rate items to develop and enhance the model. The number of ratings submitted by a user determines the precision of a recommendation. The disadvantage of this method is that it needs user effort for data. Newcoming users are not always willing to provide sufficient information.

Implicit feedback is feedback that is not inputted directly to the system.The system automatically collects information about the user's preferences based on the user's purchase history, navigation history, time spent on certain web pages, links followed by the user, e-mail content, and button clicks, among other actions. Therefore, this process eliminates the problem suffered by explicit feedback as it decreases the workload for users by inferring their preferences based on their interaction with the system. The disadvantage of this method however, although the approach requires minimal human involvement, it is less accurate.

There are a variety of ways to implement a recommendation system. Although there are an infinite number of ways to construct the system, the majority can be separated into distinct types. According to the publication [], there are three standard approaches in the literature. "Fig. 1" depicts the various types of available techniques. Content-based filtering, collaborative filtering, and a combination of the two (hybrid) are the approaches. The paper is especially interested in model-based collaborative filtering approaches, as deep learning falls within this category.

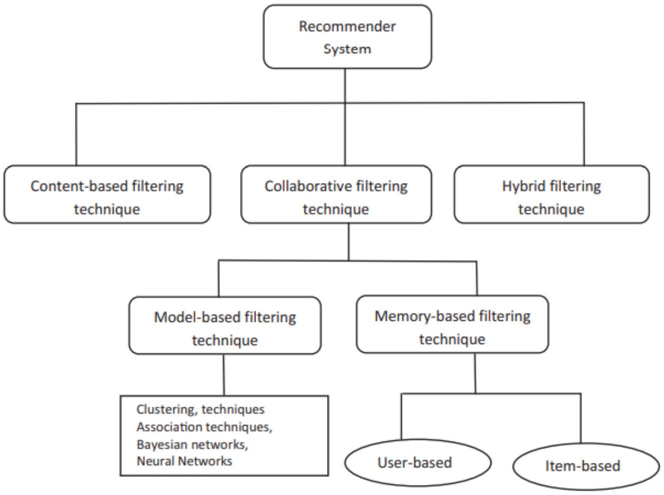

Recommender System
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based filtering technique
Memory-based filtering technique
Clustering, techniques
Association techniques,
Bayesian networks,
Neural Networks
User-based
Item-based

Fig. 1. Recommendation system categories

Content-based filtering algorithms.In order to make predictions, content-based approach is a domain-dependent algorithm that places greater emphasis on the study of item properties. When recommending documents such as web pages, magazines, and news, content-based filtering is the most effective method. In the content-based filtering technique, recommendations are generated based on user profiles utilising characteristics gleaned from the content of the things the user has previously evaluated. The user is recommended things that are primarily connected to the positively scored items.

Collaborative filtering algorithms suggest items based on the preferences of numerous users. Given previous interactions between users and goods, recommender algorithms learn to predict future engagement based on the similarity of user preferences. These recommender systems construct a model based on a user's past actions, such as things previously purchased, ratings given to those items, and judgements made by other users that are comparable. If a group of individuals have made similar judgements and purchases in the past, such as selecting a movie, there is a strong likelihood that they will agree on additional future choices. For instance, if a collaborative filtering recommender determines that you and another user have similar movie preferences, it may suggest a film that it knows this other user already like.

III. Fundamentals of Deep Learning

In this section, the paper presents some fundamental concepts on Deep Learning. The goal is to prepare the reader for the upcoming sections which will rely on the methodology of deep learning and neural networks. First, the section introduces the reader to deep learning. then the paper proceeds with an explanation on the architecture of neural networks starting with the basic unit which is a neuron. lastly, the learning process is explained.

A. Introduction to Deep Learning

Neural networks or deep learning is a subset of machine learning. It is a specified algorithm in which to solve machine learning problems. Some examples of said problems are image recognition, regression tasks, Natural language processing and much more. To have an idea of the motivation behind neural networks, it is worth looking at the history behind it.

Perhaps the most influential works that has been done to the study of neural networks originated from the years 1950s and 1960s by scientist Frank Rosenblatt where he developed the artificial neuron called the perceptron. [1] He was inspired by Warren McCulloch and Walter Pitts earlier study of the development of a mathematical model based on a biological neuron which can be seen in "Fig. 2". [2]

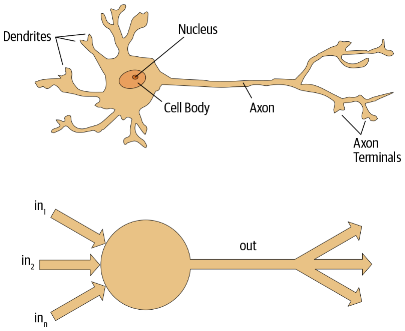


Fig. 2. Biological neuron and its mathematical model. [2]

To build on the intuition of neural networks, the formal definition of a neural network is best explained by [3] where the paper states that ''A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns." [3] To summarize, the key important elements that can be found in a neural network are individual nodes often referred as neurons that are connected to form a network, a set of weights that adjusts during an adaptation or learning process to produce the right outputs.

B. Neurons

Before going into the general structure of a neural network, one must understand the building blocks of an artificial neural network. The basic building block of an artificial neural network is called an artificial neuron and its mathematical model is depicted in "Fig. 3" [4]. The model compromises of multiplication, summation, and activation operations. At the input part of the neuron, every input is multiplied with a weight which signifies the importance of the inputs. This means the inputs are weighted. The neuron itself will sum up all the inputs that were multiplied by the weights and a bias term. Lastly, at the output stage, the resultant sum will be passed through an activation function. it decides whether a neuron is activated or not and produces a prediction value. [5].

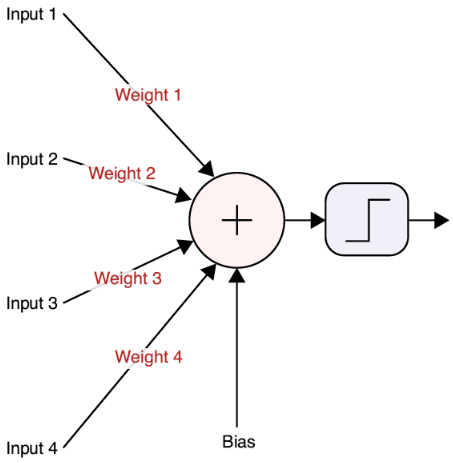


Fig. 3. Artificial Neuron. [4]

C. Neural network architecture

With the basic building block in place, it is now the time to observe a basic neural network architecture. The most common and simple architecture that best explains the basics of a neural network is known as a multilayer perceptron (MLP). “Fig. 4” shows the architecture of the MLP. As observed, there are three layers involved in the model. The first layer is called the input layer. The second layer is what we call the hidden layer, and lastly, the third layer is the output layer. There can exist an infinite number of hidden layers which will add to the abstraction level of features to be learnt. This is usually not the case as there is a threshold where too many layers can incite a problem known as overfitting. Notice in “Fig. 4”, all neurons are connected with other neurons at the adjacent layer. This is called fully connected or dense. The term deep learning refers to the many hidden layers involved making the model ‘deep’. [6]

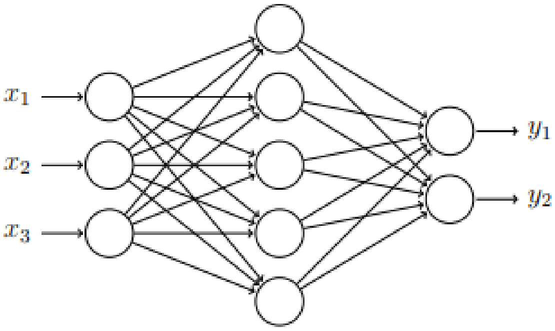


Fig. 4. Architecture of MLP.

The learning process of the model is perhaps the most crucial part to explain how neural networks can produce the output we want based on the inputs. In deep learning, there exists many types of learning methods such as supervised learning and unsupervised learning. Supervised learning is learning with pre-labelled inputs, which act as targets. The output is then compared with the target value, and the difference between the values are calculated. [6] This is achieved by utilising a loss function. A common loss function is called Mean Squared Error (MSE) and is defined as follows

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{1}$$

After the loss is calculated, we perform backpropagation to update the weights of the model. To get the new updated weights, an optimizer is usually deployed before backpropagation. The simplest optimizer is known as gradient descent. For every iteration, the optimizer will calculate a new gradient to minimise the loss function. For that, we will need a learning rate to adjust how much step the optimizer will take. That is the whole process of a neural network learning to produce the desired output. [6]

IV. Implementation of a Movie Recommendation System

In this section of the seminar, the paper details the implementation part that has been done. I have chosen to implement a movie recommendation system that gives out a suggested list of movies for the user. The section will begin with the framework that has been used to realise the experiment. For example, which libraries that was used for the experimental part with includes TensorFlow and the Microsoft recommenders library. Then, the paper goes in depth into explaining the model architecture that I have adopted for the implementation. Furthermore, this section includes even more details in the implementation like the explanation on the dataset that was used and how the dataset is prepared to feed to the model for the training part for the model. Lastly, this section provides a graphical figure as a result of the training of the model.

A. Framework Used For Implementation

Before continuing to the implementation, it is wise for one to get an understanding of the software libraries used in the project. For the programming part of the implementation, most if not all the work that was done was achieved by using the TensorFlow library. TensorFlow is an open-source framework for developing and deploying machine learning models. It incorporates many of the standard algorithms and patterns required for machine learning. This allows the user to focus on solving some specific machine learning problems without having to master the underlying math and logic. [7] TensorFlow includes many modules and functions that speed up the process of implementation as it provides high level application programming interface (API). On top of TensorFlow, there exists another module that is handy in the design and development of neural networks, which is keras. Together, both provided very useful functions for example, tf.keras.models to build some layers for the neural network, optimizers like adam optimizer, loss functions model.fit to train and many more.

The next library that was influential in this implementation is the Microsoft's recommender library. The said library contains functions to simplify common tasks used when developing and evaluating recommender systems. The examples and modules in the library are provided as Jupyter notebooks. The examples detail some learnings putting emphasis on five key tasks which are to prepare data, example models, evaluation algorithms for offline metrics, Tuning and optimizing hyperparameters for recommender models and Operationalizing models in a production environment on Azure.

B. Model architecture

As mentioned before, for the implementation of a working movie recommendation system, I have chosen to realise the model that is based on the paper [8]. The name of the generalised framework model is NCF which stands for neural collaborative filtering. As seen in "Fig. 5", there are total of four components which makes the architectural design of the model. The first layer is the input layer which converts both user and item data to representational vectors in an embedding space respectively. The inputs are then passed to two parts which are a GMF layer, and a MLP component. The GMF layer performs inner product of user and item vectors. On the other hand, the MLP is a multi-layered neural network which consists of multiple layers to learn the latent factors and map them to prediction scores. The output to both components is then feed into a final layer which is represented as NeuMF which stands for Neural Matrix Factorization. The last layer concatenates the results from previous layers and finally output a prediction confined by a sigmoid function as its activation function.

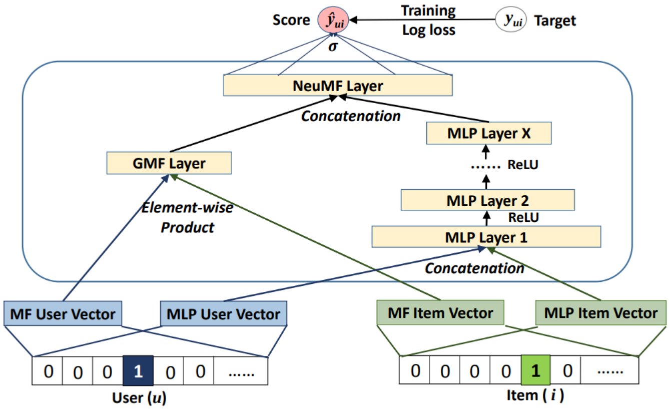


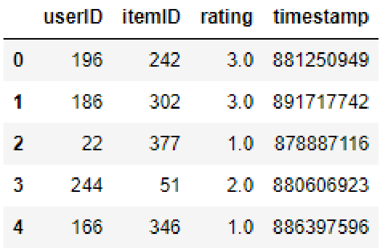
Fig. 5. NCF architecture. [8]

C. Dataset description

The dataset that was used during the training phase of the project was the Movielens dataset developed by the GroupLens Research Project. The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. Based on the description of their website and their corresponding published paper [9], this data set consists of 100,000 ratings (1-5) from 943 users on 1682 movies. Each user has rated at least 20 movies. Simple demographic info for the users (age, gender, occupation, zip). According to the paper [8], This movie rating dataset has been widely used to evaluate collaborative filtering algorithms.

D. Dataset preparation

With the dataset in place, I turned my attention to preparing the dataset to be fed to the model to train. A sample of the Movielens-100k dataset can be seen in "Fig. 6" . The first column depicts userID which represents the unique user that rated, the second column depicts itemID that shows the movie followed by its rating column that is a value from 1 to 5 and timestamp of when the rating is given. There exists more information provided by the dataset like demographic information of the users, but for this implementation, I only focused on the data that is in "Fig. 6" described as columns in this paper.



	userID	itemID	rating	timestamp
0	196	242	3.0	881250949
1	186	302	3.0	891717742
2	22	377	1.0	878887116
3	244	51	2.0	880606923
4	166	346	1.0	886397596

Fig. 6. Dataset sample.

Before training the model, the data set was modified to transform explicit feedback, which was the rating column, into implicit feedback. This means that every movie with which the user has interacted is allocated the number 1, whereas movies with which the user has not interacted are awarded the value 0. This method is known as negative sampling. For the train-test split, the data have been chronologically split using the function " python chrono split". The actual breakdown is 75 percent for training and 25 percent for testing. Then, I remove from the test set any users or items that do not appear in the training set. In addition, a new test set is produced for a distinct evaluation process. All evaluation techniques will be given in the section titled "Results analysis" which follows. The datasets are then saved as CSV files.

E. Training Phase

For the project's training phase, I chose to adopt the concept stated in the paper [8]. Among the specifics of the training section is the loss function. The applied loss function was binary cross-entropy loss. This is known as log loss in the library. Additional parameters include the optimizer, number of latent components, and batch size. The Adam algorithm, an adaptation of the standard Stochastic Gradient Descent (SGD), is utilised for the optimizer. This is because the Adam optimizer adapts the learning rate for each parameter by executing smaller updates for often used parameters and bigger updates for infrequently accessed parameters. In addition to producing faster convergence for both models than the standard SGD, this technique eliminates the need for tweaking the learning rate. [8] Therefore, the utilised learning rate was 1e-3.

In order to construct these models, the library function NCF is employed. The function expects numerous arguments, including the data to be utilised, the model type (GMF, MLP, or NeuMF), the number of latent factors, the size of the model's layers, the number of epochs, the batch size for training, and the learning rate. All of the previously listed parameters must be given when building models.

The model was trained for 100 epochs with a batch size of 256 for each iteration. In the recommenders' library, the number of latent factors is known as n factors. This parameter regulates the dimension of the latent space. It is significant since it controls the accuracy of the training set predictions. The quality of said predictions improves as the n factors increases. To start the training, the latent factors were initialized to a value of 16 as it shows the best middle ground for performance and prevention of overfitting. The actual training is done by calling the model.fit method.

After training the model for 100 epochs, the graphical representation of the number of epochs trained and the models training loss is recorded. "Fig. 7" depicts the training graph of the model. The final training loss recorded after 100 iterations is 0.164153.

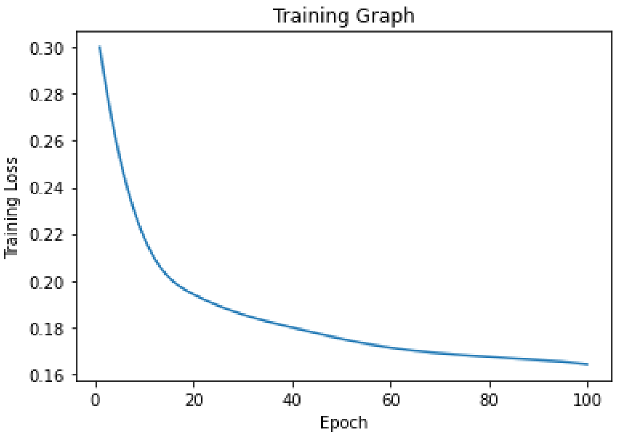
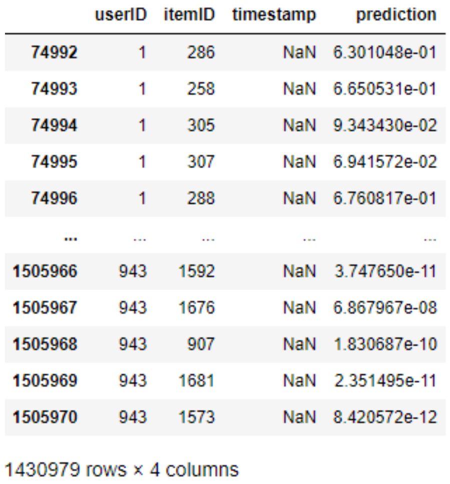


Fig. 7. Training graph.

In the next section, this seminar paper describes the evaluation protocols that was used to measure the performance of the model. It details on the description of each evaluation metrics used and why it has been utilised.

V. Analysis of Results

In this section of the paper, the reader will have an insight into how the evaluation process is conducted during the experimental phase. After the training phase, there is the possibility to output the recommendation list by using the function. "Fig. 8" depicts a sample of the final list of recommended items along with the prediction of whether the user will interact with the item. Although the model has given a list of recommended movies, it is near impossible to know the performance of the recommendation system by looking at the list alone. Therefore, for the evaluation process, it is required to know some offline evaluation metrics that are suitable for analysing the performance of the model

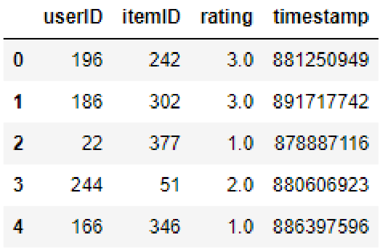


	userID	itemID	timestamp	prediction
74992	1	286	NaN	6.301048e-01
74993	1	258	NaN	6.650531e-01
74994	1	305	NaN	9.343430e-02
74995	1	307	NaN	6.941572e-02
74996	1	288	NaN	6.760817e-01
...	...	...	...	...
1505966	943	1592	NaN	3.747650e-11
1505967	943	1676	NaN	6.867967e-08
1505968	943	907	NaN	1.830687e-10
1505969	943	1681	NaN	2.351495e-11
1505970	943	1573	NaN	8.420572e-12

1430979 rows × 4 columns

Fig. 8. prediction list.

The offline assessment criteria selected for analysis in this experiment are based on conventional evaluation methods contained in the Microsoft recommenders' library. Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), Persicion@K, and Recall@K are the metrics included. MAP is a measure of how many of the recommended items are present in the collection of authentically relevant documents. This method takes into account the order of the recommendations. NDCG at k is a measure of how many of the first k recommended items averaged across all users are in the set of truly relevant content. Unlike precision at k, this metric considers the sequence of the recommendations. Precision at k is a measure of how many of the first k recommended documents averaged across all users are in the collection of truly relevant documents. This measure does not account the order of the recommendations.



	userID	itemID	rating	timestamp
0	196	242	3.0	881250949
1	186	302	3.0	891717742
2	22	377	1.0	878887116
3	244	51	2.0	880606923
4	166	346	1.0	886397596

Fig. 9. Timing graph for pipeline analogy.

The table displays the evaluation metric values. The model's

MAP value has increased by 0.045272. Meanwhile, the recorded NDCG value is 0.193614. Other than that, the result produced for Percision@K is 0.175822. Recall@K's final value is 0.099722.

VI. Conclusion

To conclude this seminar, the paper summarises what has been presented. Firstly, a introduction to recommendation systems and deep learning was given. To enrich the reader's knowledge, and to prepare for the implementation part, an overview of recommendation systems and also some fundamental knowledge on deep learning was presented. Then, an implementation of a movie recommendation system was explained in detail including the analysis of results after training the deep learning model.

References

[1] Nielsen, M. Neural networks and deep learning. (Determination Press,2015)

[2] Howard, J. & Gugger, S. Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD. (O'Reilly Media, Incorporated,2020), https://books.google.no/books?id=xd6LxgEACAAJ

[3] Gurney, K. An introduction to neural networks. (CRC Press,2014)

[4] Glassner, A. Deep Learning: A Visual Approach. (No Starch Press,2021,6)

[5] Suzuki, K. Artificial neural networks: methodological advances and biomedical applications. (BoD–Books on Demand,2011)

[6] Hernández Fuster, O. Self-driving RC car using deep learning. (2021)

[7] Moroney, L. AI and Machine Learning for Coders a programmer's Guide to Artificial Intelligence. (O'Reilly,2021)

[8] 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)

[9] Harper, F. & Konstan, J. The movielens datasets: History and context. *Acm Transactions On Interactive Intelligent Systems (tiis)*. **5**, 1-19 (2015)