UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COLLABORATIVE FILTERING UTILIZING NEURAL NETWORKS

Diploma Thesis

Vasiliki Karamoustou

Supervisor: Michael Vassilakopoulos

December 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COLLABORATIVE FILTERING UTILIZING NEURAL NETWORKS

Diploma Thesis

Vasiliki Karamoustou

Supervisor: Michael Vassilakopoulos

December 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΣΥΝΕΡΓΑΤΙΚΟ ΦΙΛΤΡΑΡΙΣΜΑ ΜΕ ΧΡΗΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

Βασιλική Καραμούστου

Επιβλέπων: Μιχαήλ Βασιλακόπουλος

Δεκέμβριος 2022

Approved by the Examination Committee:

Supervisor **Michael Vassilakopoulos**
Chairman of Department and Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Panagiota Tsompanopoulou**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Eleni Tousidou**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

Above all, I would like to express my deepest gratitude to my supervisor, Chairman of the Department, and Professor Mr. Michael Vassilakopoulos for his consistent support and guidance during the running of this project and the writing process of this thesis.

Besides my supervisor, I wish to express my sincere appreciation to Associate Professor Ms. Panagiota Tsompanopoulou and Laboratory Teaching Staff Ms. Eleni Tousidou who accepted to be members of the examination committee.

Furthermore, I want to extend my recognition to Ph.D. Candidate Mr. Vaios Stergiopoulos and the rest of the research team for their effort during data collection, a piece of which I used in my project.

Last but certainly not least, I feel grateful to my family and friends for supporting me throughout my studies and specifically during the development of this thesis.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Vasiliki Karamoustou

Diploma Thesis

COLLABORATIVE FILTERING UTILIZING NEURAL NETWORKS

Vasiliki Karamoustou

Abstract

It is reasonable to state that the majority of people interact with recommendation systems (RS) on a daily basis. The rapid growth of data has led users to be confronted with a plethora of options which despite their multiple benefits can create uncertainty as to which one is more suitable to serve one's purpose. People have become extremely demanding and as a result, the role of RS is vital. Amazon, YouTube, and Netflix are just some of the applications that have implemented RS to provide a more trouble-free experience for their users.

There are two main categories of RS, Content-Based Filtering(CBF) and Collaborative Filtering (CF). The main difference between the previously stated categories is the type of information, based on which they make recommendations. Additionally, there are hybrid systems, which manage to combine the previous two approaches. With that being stated, for the purpose of this thesis, we will be focusing on CF RS.

In this work, we will get into RS's operation, hardships, and how Machine Learning(ML), and specifically Neural Networks(NN), can be applied in such systems. To specify, this thesis attempts to integrate two different networks based on the Neural Collaborative Filtering (NCF) framework. On one hand, a Matrix Factorization (MF) network is used and on the other two different NNs, a Multilayer Perceptron (MLP) or a Long Short-Term Memory (LSTM). Eventually, after experiments on two different datasets, it is certain that the models have similar scores with LSTM having slightly more accurate results. In detail, firstly we have used the ”Book-Crossing” dataset and secondly data from AMiner Citation Network.

Keywords:

Recommendation systems, Collaborative Filtering, Neural Networks, Matrix Factorization, Multilayer Perceptron, Long Short-Term Memory

Διπλωματική Εργασία

ΣΥΝΕΡΓΑΤΙΚΟ ΦΙΛΤΡΑΡΙΣΜΑ ΜΕ ΧΡΗΣΗ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

Βασιλική Καραμούστου

Περίληψη

Στη σημερινή εποχή, πιθανώς όλοι μας αλληλοεπιδρούμε καθημερινά με Συστήματα Συστάσεων (ΣΣ). Λόγω της ραγδαίας αύξησης των δεδομένων, οι χρήστες έρχονται αντιμέτωποι με μια πληθώρα επιλογών. Αυτό φυσικά μπορεί να θεωρηθεί ως κάτι θετικό, όμως μερικές φορές όταν έχουμε πάρα πολλές επιλογές, μπορεί να δυσκολευτούμε στη τελική λήψη των αποφάσεων. Οι άνθρωποι γίνονται όλο και πιο απαιτητικοί, και τα ΣΣ είναι πλέον απαραίτητα. Το Amazon, το YouTube και το Netflix είναι μόλις μερικές από τις εφαρμογές που ήδη κάνουν χρήση των ΣΣ με σκοπό να γίνουν πιο φιλικά προς τον χρήστη.

Υπάρχουν δύο κύριες κατηγορίες ΣΣ, το Βασισμένο στο Περιεχόμενο (Content Based-CB) σύστημα και το Συνεργατικό Φιλτράρισμα (Collaborative Filtering-CF). Η κύρια διαφορά μεταξύ τους είναι το είδος των πληροφοριών, βάσει του οποίου πραγματοποιούν συστάσεις. Φυσικά, υπάρχουν και τα υβριδικά ΣΣ, τα οποία συνδυάζουν τις προηγούμενες δύο προσεγγίσεις. Για τις ανάγκες αυτής της Διπλωματικής Εργασίας, θα επικεντρωθούμε στα ΣΣ με Συνεργατικό Φιλτράρισμα.

Σε αυτή την εργασία, θα ασχοληθούμε με την λειτουργία των ΣΣ, τις δυσκολίες αυτών, καθώς και την εφαρμογή Μηχανικής Μάθησης, και πιό συγκεκριμένα Νευρωνικών Δικτύων (ΝΔ) σε αυτά τα συστήματα. Πιό αναλυτικά, χρησιμοποιούνται συνδυαστικά δύο διαφορετικά δίκτυα βάσει του «Neural Collaborative Filtering (NCF)» πλαισίου, όπως αυτό αποτυπώνεται στο άρθρο. Από τη μία πλευρά, χρησιμοποιείται ένα δίκτυο με Παραγοντοποίηση Μήτρας (MF) και από την άλλη δύο διαφορετικά είδη ΝΔ, ένα Πολυστρωματικό Αντίληπτρο (MLP) ή ένα Δίκτυο Μακράς Βραχύχρονης Μνήμης (LSTM). Τέλος, μετά από πειράματα σε δύο διαφορετικά σύνολα δεδομένων καθίσταται βέβαιο ότι τα δύο μοντέλα έχουν παρόμοιες επιδόδεις με το Δίκτυο Μακράς Βραχύχρονης Μνήμης να ενισχύει ελαφρώς τα τελικά αποτέλεσματα. Πιο αναλυτικά, χρησιμοποιήσαμε το ”Book-Crossing” σύνολο δεδομένων καθώς και δεδομένα από τον παγκόσμιο ιστό και συγκεκριμένα από την Aminer βιβλιοθήκη.

Λέξεις-κλειδιά:

Συστήματα Συστάσεων, Συνεργατικό Φιλτράρισμα, Νευρωνικά Δίκτυα, Παραγοντοποίηση Μήτρας, Πολυστρωματικό Αντίληπτρο, Δίκτυο Μακράς Βραχύχρονης Μνήμης

Table of contents

List of figures

List of tables

Abbreviations

AI	Artificial Intelligence
RS	Recommendation systems
CF	Collaborative Filtering
CBF	Content Based Filtering
NN	Neural Networks
MF	Matrix Factorization
MLP	Multilayer Perceptron
LSTM	Long Short-Term Memory
SVD	Singular Value Decomposition
DL	Deep Learning
NCF	Neural Collaborative Filtering
etc.	Etcetera
sb	Somebody
smth	Something
HP	Hyper-parameters

Chapter 1

Introduction

Finding a variety of relevant sources to our research needs, all in one place, is one of today's highest trending topics. Information overload has been extremely apparent during our days and for that reason, most online services have adopted RS. RS are software tools that generate and provide users with suggestions about entities they are interested in by utilizing various strategies. A tremendous amount of research has been done on RS, mostly related to MF methods. However, against the vast number of papers on MF, there is relatively little work based on using Deep Neural Networks for recommendation.

Recommendations provided by computing devices, in contrast to the ones made by humans, can consider vast amounts of data, or even hidden knowledge extracted in an indirect way. For that reason, RS are used in a variety of areas, commonly related to movies, music, advertisements, friend suggestions on social media, and so forth. However, there is no such convenience for researchers to dig into all scientific documents of their interest. And even if there are a few platforms, such as SpringerLink, which provide easy access to articles, eBooks, and other resources, most of their requirements are impractical. For example, these recommendation platforms require denoting the field of interest in an explicit manner.

1.1 Motivation

This thesis is based on the NCF framework presented in [11]. The model proposed in the paper implements two different networks that are later combined to make the final suggestion. On one hand, we have a standard MF network and on the other a MLP network. The essential idea is taking advantage of both the linearity and non-linearity provided by the two networks mentioned. Additionally, this work explores the use of a LSTM network for learning the function from data, rather than the MLP network that has been used by the work previously mentioned.

The fact that the domain of book or citation recommendation has only been reviewed by a minute number of research studies, has been one of the primary reasons for which we decided to deal with two datasets about scientific papers and books. Lastly, the thesis discusses and analyzes the evaluation metrics from extensive experiments on the datasets by the explored models. At the same time, tuning of different hyper-parameters is taking place due to optimization of validation results.

1.2 Contributions

In this thesis, we study the book and citation recommendation problem. Moreover, in order to alleviate this problem we integrate CF methods and Neural Network architectures. Our major focus is to provide the contributions summarized below.

1. Review and study of the related work and obtained methods utilizing RS.

2. Mention and explain preliminary knowledge of DL models and MF.

3. Demonstrate the two datasets used and analyze which information each of them comprises.

4. Describe the preprocessing techniques applied to the datasets.

5. Study CF systems of standard MF networks for modeling user–item interactions.

6. Propose a new approach of CF which takes advantage of NNs, named NCF.

7. Present that NCF is a generalization of MF.

8. Integrate state-of-the-art MF models and MLP or LSTM networks to achieve better results.

9. Introduce the evaluating metrics used for the judgment of the performance of the two NCF approaches.

10. Implement hyper-parameters tuning willing to improve model's training and testing outcomes.

11. Illustrate via experimental results that NCF utilizing LSTM slightly outperforms NCF utilizing MLP in two real-world datasets.

1.3 Thesis Organization

The Thesis organization is as follows. Chapter 1 was an introduction to the field of interest, explaining what this work aims to implement. The evolution of RS through the years, research relevant to them and related work are shown in Chapter 2. Secondly, Chapter 3 provides a detailed description of the essential preliminary knowledge required for the comprehension of the systems that will be implemented afterwards. Subsequently, Chapter 4 gives definite information on the datasets that are used in this thesis. Chapter 5 introduces NCF,

emphasizing on MF improvement with neural techniques. Next, Chapter 6 presents the different useful evaluation metrics for interpreting model's performances. The overall clarification and comparison of the experimental results as well as a comprehensive description of the implementation of the models are discussed in Chapter 7. Chapter 8 accommodates a brief presentation of the tools employed for the programming work. Finally, this study draws some conclusions and gives future orientations in Chapter 9.

Chapter 2

Related Work

A great amount of related studies and ongoing research have been conducted in the field of RS. Taneja and Arora in [1] have implemented a highly assistive work for briefing researchers interested in this specific domain. The study presents a synopsis of RS evolution and methodically analyzes the user-item interactions. Their extensive research mainly comprises 258 survey papers. Figure 2.1 depicts the publications related to recommendation systems in a timeline from 2002 to 2017. This periodic illustration contains the most popular and predominantly accessible papers relevant to the field of interest. The first interest to compose and publish a related article has been introduced by Robin Burke in 2002 [15]. This paper observes the aspect of hybrid RS and introduces an innovative model named EntreeC, which combines content-based information and collaborative filtering for recommending restaurants. Furthermore, it is compelling to observe that the interest and therefore the authorship of associated articles increased slowly until 2011. Since then, the domain of recommendation has entered a period of rapid growth. It is worth noticing that this surge coincides with the rapid rise of information on the internet and the consequential problem of information overload.

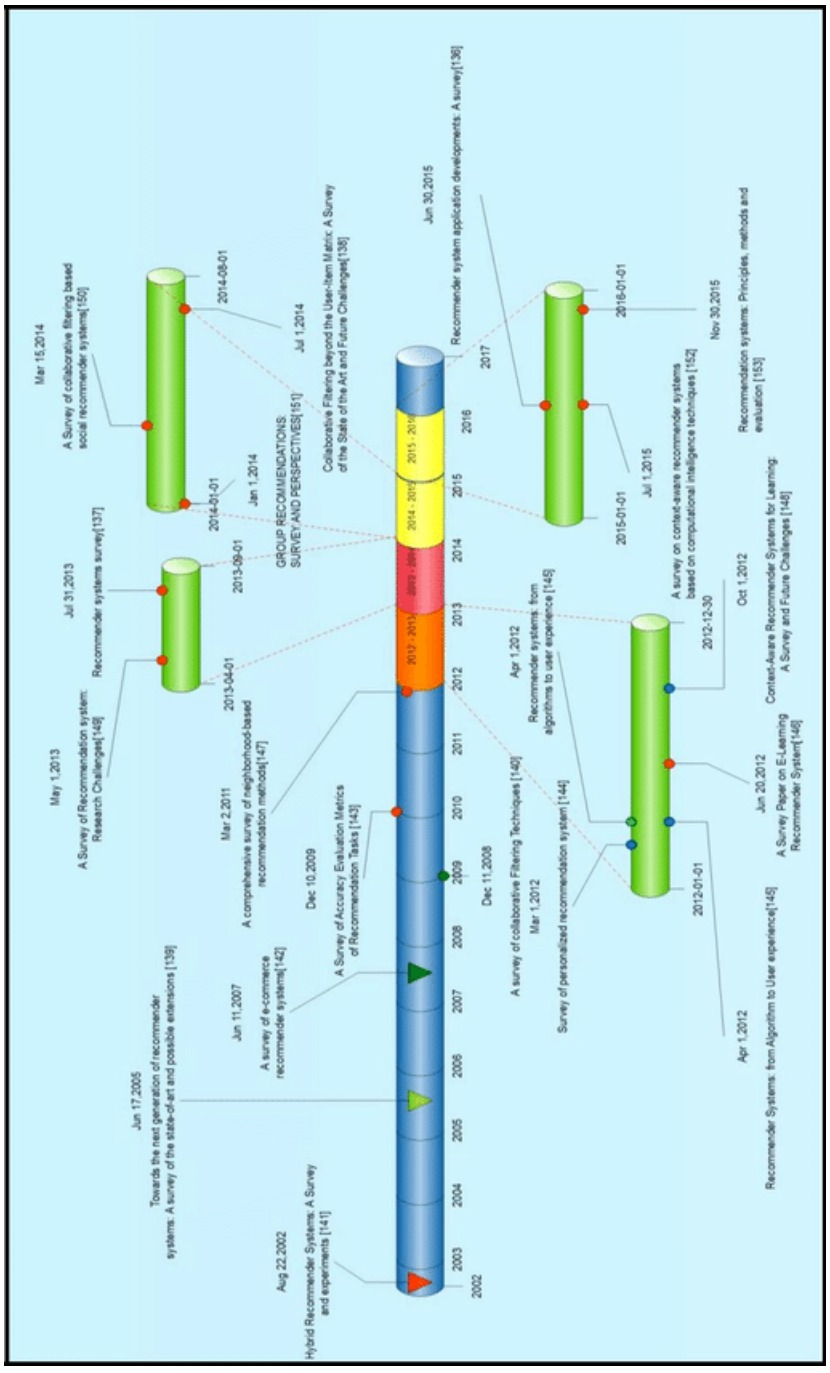

Figure 2.1: Timeline review of published recommendation systems survey papers [1]

With that being said, we will be focusing on CF for the purpose of our studies. The extended bibliography on this subject reflects its importance. CF-based methods observe and use similarities between users and items or past preferences, without using any content information to make recommendations. Paper [16] considers CF systems both in theory and in practice, via MovieLens [17]. Additionally, it discusses evaluation metrics of CF and more specifically accuracy, as the most important criteria to evaluate. Lastly, it examines security challenges to CF.

Conventionally, CF is implemented with MF due to its superior performance. Bokde et al. in [18] survey the role of MF models such us Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and Probabilistic Matrix Factorization (PMF) to deal with issues of CF algorithms. Those algorithms are often confronted with a very sparse rating matrix which causes performance degradation.

Authors of [19] and [20] have proposed a new method named WRMF (weighted matrix factorization). They manage, this way, to decrease the impact of negative ratings to the basic MF technique, by including weights in the procedure that calculates the error.

However, recently attention is increasingly shifting towards new ways to perform CF with DL techniques such as NN. Moreover, Wang et al. in [21] propose a Hierarchical Bayesian model called collaborative deep learning (CDL). This approach performs DL representation for the content information, along with CF for the ratings.

Furthermore, our approach is inspired by the early pioneer work, titled Neural Collaborative Filtering [11], which describes the approach to perform CF using NN. He et al. form a general framework named Neural Collaborative Filtering-NCF (short for NN based CF), by replacing the dot product with a neural architecture. They study the user–item interactions in three different ways, GMF (Generalized Matrix Factorization), MLP (Multi-Layer Perceptron) and NeuMF (Neural matrix factorization). Implementation of experiments on two real-world datasets (MovieLens and Pinterest) proves that NCF framework outperforms the state-of-the-art methods. In this thesis we consider that study [11] as a guideline for developing a new deep learning method which includes an LSTM network for recommendations.

Lastly, the article [22] was fundamental in providing this work the inspiration needed to carry out the HP tuning of the NCF systems during the experimental phase. This paper performs tuning of the activation function, weight initializers and epochs of train of an ANN called CATA++. Hp optimization achived to boost CATA++'s performance up to 44.2%, while de-

creasing training time. Experiments were conducted on two datasets (CiteULike and AMiner Citation Network), recording Recall and nDCG metrics.

Chapter 3

Preliminaries

3.1 Deep Learning

Machine learning and deep learning are the cornerstones of the upcoming revolution in computing. DL (or Neural Networks) can be considered as a subfield of ML which imitates how the human brain thinks and learns. Concerned with computer algorithms for data processing and pattern recognition, manages to learn and improve on its own. [23] As anyone can understand, this is a compelling feature that offers great promise for recommender systems. NN's goal is developing robust algorithms that can be used to model common problems such as making accurate recommendations.They have the ability to learn from a set of training data and improve their accuracy over time. The fundamental components of NN are called artificial neurons or nodes. Those are simple computational units with weighted input signals who produce an output signal using an activation function. Activation function, as implied by its name, decides whether a neuron should be activated or not. Typically, nodes are aggregated into layers. A network can have multiple layers. The bottom layer, which gets its input from the dataset is called the visible layer. Layers after that are called hidden layers and the final hidden layer is called output layer. Output layer is responsible for producing the final output.

Machine Learning Vs Deep Learning

Deep learning is a subset of ML. The hierarchy follows the following sequence. At the top is AI. A subfield of AI is ML. Beneath that, are NNs that basically make up the backbone of DL. ML algorithms leverage structured, labeled data to make predictions.

- The first difference is relevant to their structure. Conventional ML models like Linear Regression (LR) are constructed really simply. On the other hand, NN can have different combinations of layers which leads us to the conclusion that those networks are compounded.
- Additionally, as shown in Figure 3.1, deep learning models automate the process of making predictions, while they learn from their own errors. Contrariwise, an engineer who builds up a ML algorithm is also obliged to select manually a few parameters. Also, a classifier is qualified to examine the error and readjust the algorithm if it has a higher value than it should.
- Last but not least, in order to operate accurately a DL algorithm needs significantly more data than ML algorithms. This is due to the intricate structure of multiple layers that NNs comprise.

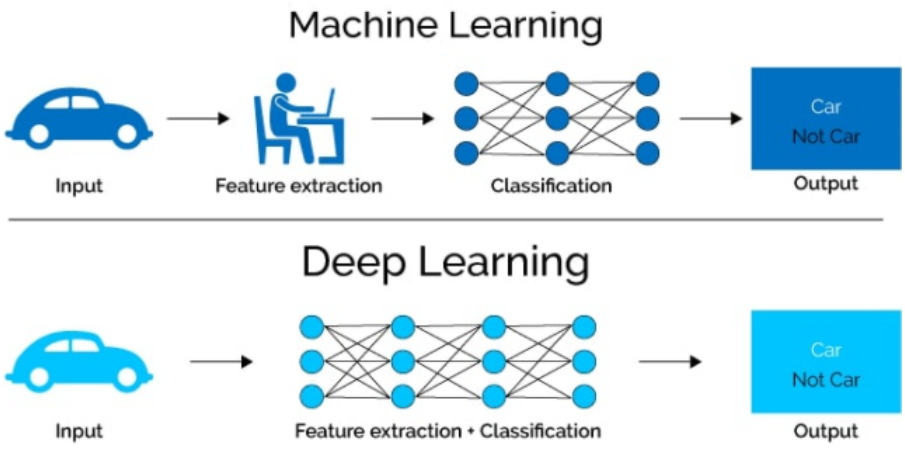

Figure 3.1: ML vs DL [2]

3.1.1 Multilayer Perceptron or MLP

A common question is how scientists got computers to think in a similar way to people, an operation renowned as Artificial intelligence (AI). One of the major concepts behind this achievement is the MLP [24]. As a preparatory step, merely a perceptron is going to be explained. Perceptron is heavily inspired by our brain's most basic unit of thinking, which is the neuron. Each neuron has a nucleus, which is nothing but a kernel that takes in inputs from other neurons and gives out outputs to other neurons. Likewise, a perceptron is constructed of three basic components. A function, which is the reasoning kernel of the perceptron, the inputs that come in from other perceptrons and the set of outputs that go out from it. Further, neurons are never alone but densely connected in big groups, cooperating in concert to deliver the desired outcome. Perceptron's organization mimics the way that human neurons are physically organized. Perceptrons are organized in layers all connected and feed of each other's inputs and outputs. Accordingly, this is where the multilayer part of MLP emerges. For multilayer networks the output of one layer becomes the input to the next layer.

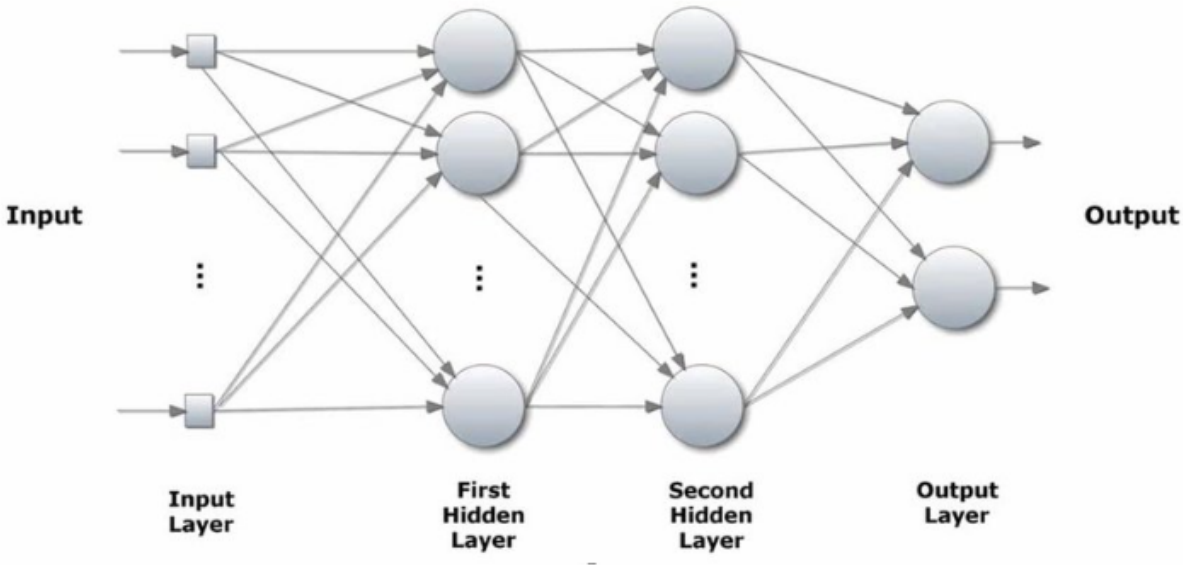

Figure 3.2: Multilayer Perceptron's Architecture [3]

The equation that describes the above operation is as follows:

$a^{m+1} = f^{m+1}(W^{m+1} \cdot a^m + b^{m+1})$ for $m = 0, 1, ..., M - 1$ Where:

- a^{m+1} is the output of layer (m + 1)
- a^m is the output of layer m and input of layer (m + 1)
- f^{m+1} is the activation function of layer (m + 1)
- W^{m+1} and b^{m+1} is the number of layers in the network

Signal path through the nodes of each layer only flows one way and consequently we discuss a feed forward NN. MLP comprises three different types of layers as shown in Figure 3.2. The input layer, which meets the signal to be handled. The hidden layers, where the computational work is done and an output layer that fulfills the prediction task.

Subsequently, the training process of the MLP is going to be analyzed. There are three basic parts of learning 3.3.

1. First, the MLP gives an output not only based on its inputs, but also on its transfer function.

2. Oftentimes that output turns out to be wrong and must change. The changing process is called a learning algorithm. The first attempt of constructing a training algorithm for multilayer networks was presented in the work of Paul Werbos in 1974. It was not until the mid-1980s that the backpropagation algorithm was rediscovered and widely publicized. Backpropagation is the act when MLP (or any other multilayer network) goes back through its layers and improves itself in depth to such an extent that the next output is superior.

3. The operation of repeating is called an Epoch. The whole process continues until the error is at the lowest value. Every epoch that a MLP goes through brings it closer to the ideal output.

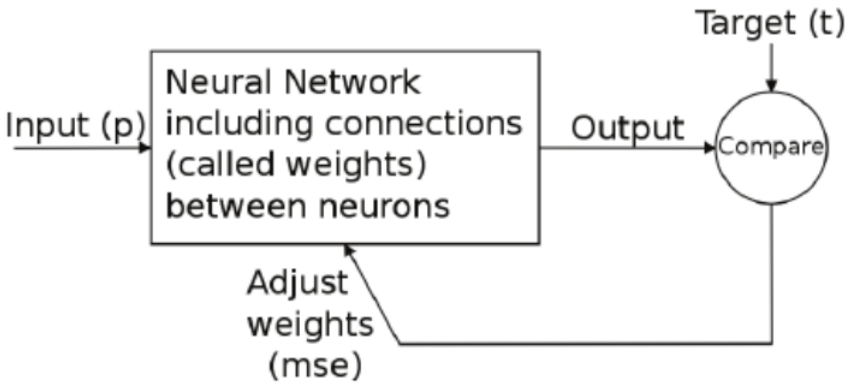

Figure 3.3: Training procedure [4]

3.1.2 Reccurent Neural Networks or RNNs

RNNs are very versatile as we can use them in a host of applications. They have connections with loops (Figure 3.4), which leads us to the result that they remember the previous information and make use of them for current input processing [25].

An issue that RNNs suffer from is called vanishing gradient problem [6]. In other words, the problem rises when gradient diminishes through backpropagation, because if it gets really small it won't significantly affect the learning process. By gradient we refer to the number used for weight upgrading.

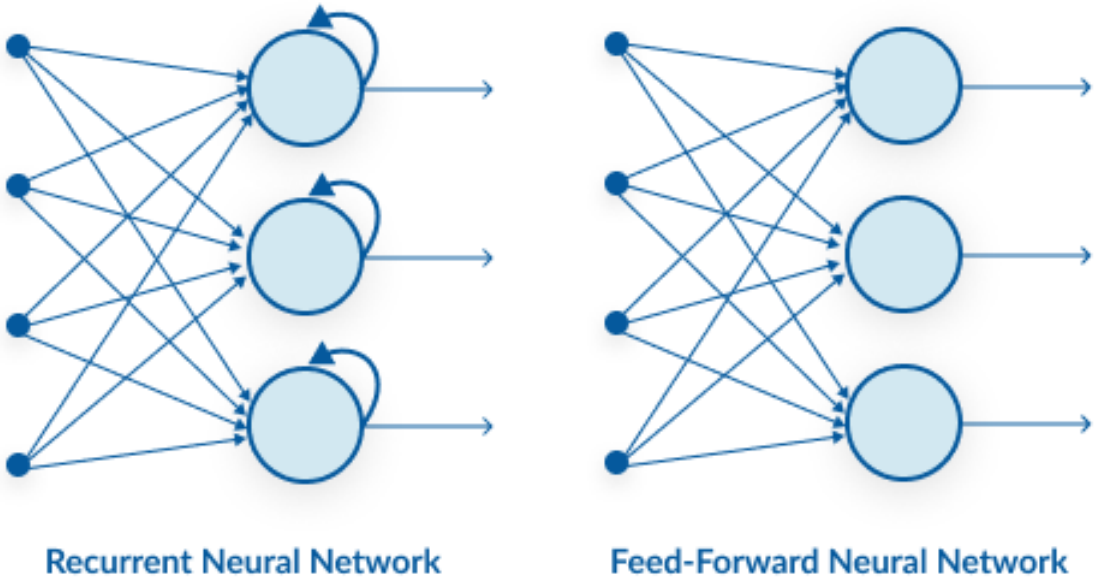

Figure 3.4: Recurrent NN's Vs Feed-Forward NN's Architectures [5]

3.1.3 Long Short-Term Memory Networks or LSTM

One of the most used RNN architectures that contributes to the solution of the vanishing gradient problem is LSTM. LSTMs are differentiated by common RNNs because of the use of memory blocks into layers. A block has components that make it smarter than a classical neuron and a memory for recent sequences. They have internal structures that can control information streams. Every hidden unit is replaced with an LSTM cell and each cell has three gates, all containing a sigmoid activation. Apart from these gates each cell has another vector that modifies, utilizing the tanh activation, the **cell state** to new values that fit better to the NN (Figure 3.5).

The three gates mentioned before, as well as their use, are shown below:

1. The **input gate** controls whether current step's information is relevant.
2. The **forget gate** controls which information should kept and which not.
3. The **output gate** controls whether the information of the current cell state must become visible to the next hidden state.

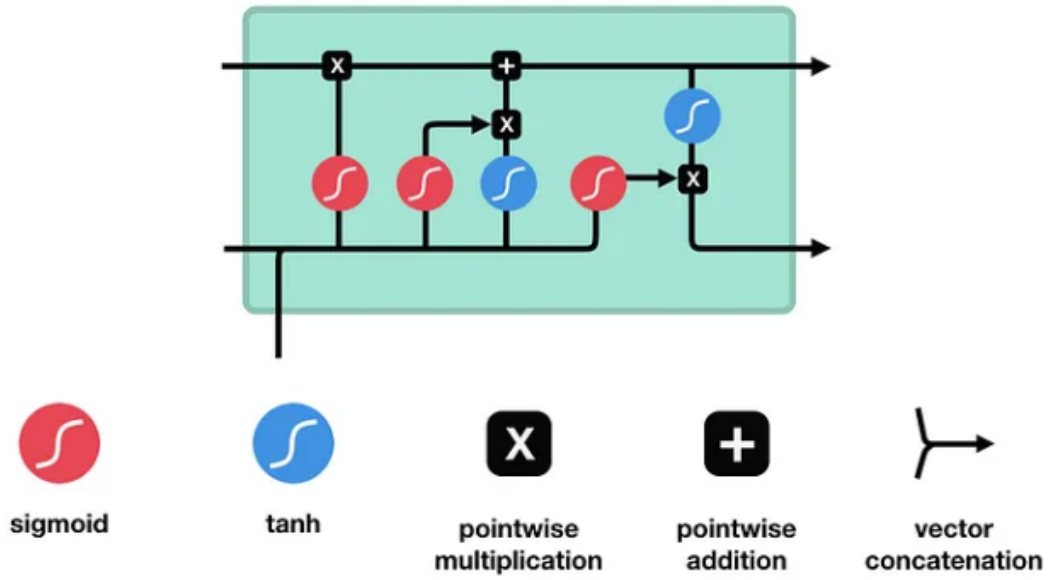

Figure 3.5: LSTM Cell and It's Operations [6]

3.2 CF's Concept

Existing methods for RS can roughly be categorized into two main approaches: the Content Based Filtering (CBF) and Collaborative Filtering (CF) based methods. The CBF method focuses on purchases users have made in the past or personal information such as gender or age. On the contrary, the CF method differs since the prediction is generated based on what similar users have purchased in the past. This leads to questioning what those similar users are and how are they determined. For example, in Figure 3.6 John and Alice are considered similar because they both have interacted with items A and C. In this case, John also purchased items B and D. So those will be recommended to Alice too. As far as one can see, CF relies on the assumption that those users who had similar choices in the past will continue to have similar choices in the future.

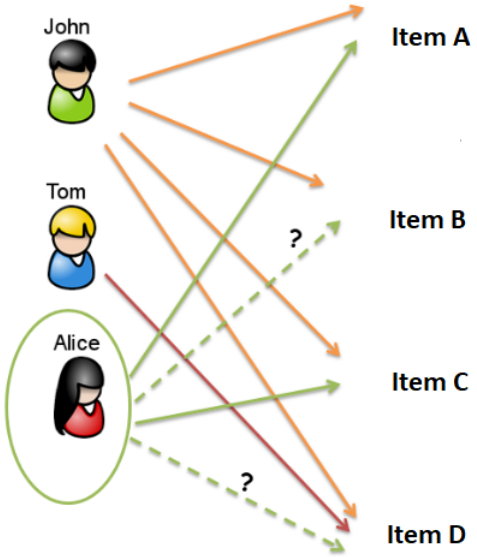

Figure 3.6: CF example [7]

3.3 Matrix Factorization

Matrix Factorization is a class of CF algorithms that are commonly used in RS. In summary, this algorithm decomposes the so-called user-item interaction matrix into a product of two matrices of lower dimensions [26, 27]. Those latent vectors are called embeddings. To increase our understanding of the performance of MF for making recommendations we are going to use an example from google developers [8].

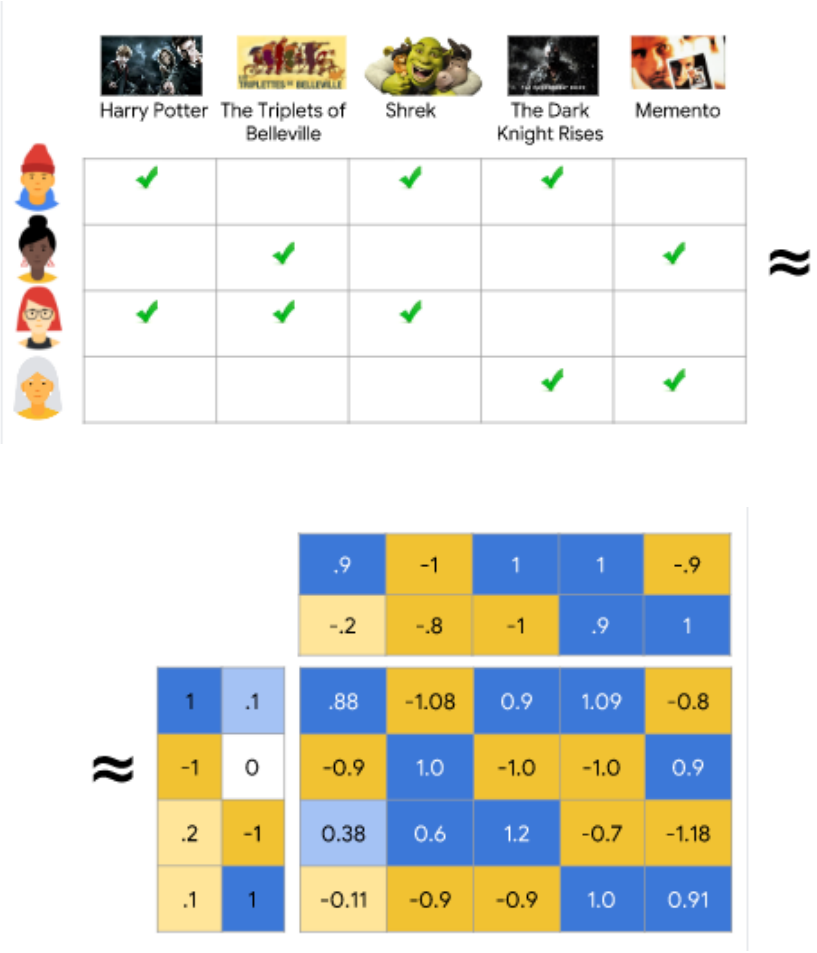

Figure 3.7: A simplified illustration of the latent vector approach, which characterizes both users and movies using two axes [8]

As mentioned, the goal of CF is to come up with embeddings. The example we are examining deals with recommending movies to users (Figure 3.7). The first vector will encapture the genre of each movie (comedy, terror, animation, etc). The second vector should represent, at some level, if the user likes each genre. Once the embeddings are prepared, the system is ready to predict whether and how much a user will enjoy a specific movie. Consequently, it will recommend to each user the most likely to enjoy movies, from those they haven't watched. The obvious question here is how we create these embeddings, and its answer is none other than MF. Generally, the result of the multiplication

of two matrices A and B will be another matrix C. The exact opposite operation is called factorization. So, factorizing matrix C means extracting matrices A and B from it. Applied to our problem, in Figure 3.7, assuming that each check mark represents a positive value and each blank space a negative one, we end up with the interaction matrix on the right side of the figures. Our goal is to make the best approximation of that interaction matrix. At this point, ML makes our purpose a lot easier while implementing matrix factorization. Therefore, after creating and training the model used for the predictions, the interaction matrix should seem like the matrix on the right side of Figure 3.7. As previously mentioned, user and movie embeddings are the ones that will be trained and the matrix we are approximating will be the dot product of them. Prior to training, these two embeddings are random which consequently leads to their product being random as well. The ML model will iteratively compare each element of the left matrix with the corresponding element of the right matrix (Figure 3.7). Additionally, following each iteration the error will be computed utilizing optimization algorithms such as Gradient Descent (GD), SVD, etc. This process will be repeated until we get a satisfactory error factor.

Chapter 4

The Data

4.1 Feedback Categories

Generally, there are two main types of feedback one can receive. Those types of feedback are known as explicit and implicit feedback.[28] Their main difference is found on whether data is provided consciously or not. Explicit data is information which includes some sort of rating. For example, explicit feedback would be if sb watches a movie and states whether liked it or not or how much. On the other hand, data can be gathered from available data streams, either directly or through analysis of explicit data. In this instance, when data has been collected via user's behavior, they are called implicit. For example, implicit feedback would be just knowing that a user watched a movie, with no concerns on rating or any other particular action. In real life situations users prefer not to spend time on rating items. Hence, most of the time people meet implicit feedback. The drawback is that implicit data are noisier and do not always have an obvious explanation.

4.2 Datasets

Substantial experiments are conducted on two real-world datasets to illustrate the efficacy of our models. This section describes the information each dataset comprises, as well as analyzes the pre-processing techniques used for preparing data for model training. One important preliminary step for implementing a reliable and effective model is data pre-processing 4.1. This term refers to any type of processing performed on raw data that transforms it to so-called "clean" data. That way data won't affect the model's reliability.

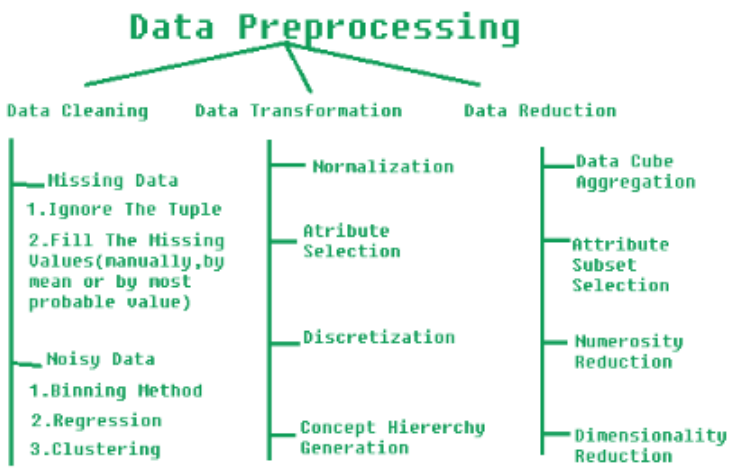

Figure 4.1: Preprocessing in ML [9].

4.2.1 Book-Crossing

Data presentation

Collected by Cai-Nicolas Ziegler, this dataset consists of book reviews. It accommodates three files. The "BX-Users" file containing user's demographic information, the "BX-Books" file which comprises details about the books and the "BX-Book-Ratings" file with the ratings each user gave to the books expressed on an explicit manner (Scaled from 1 to 10). The three tables include 278,858 anonymized users assigning 1,149,780 ratings about 271,379 books. [29] Since the focus of this thesis is on a CF approach, the interest turns to the items, the users, and what items a user interacted with. With that being said, it is obvious that the first dataset used in this thesis consists only of the "BX-Book-Ratings" table, as content-based information about users or books cannot be used from the RS.

	User-ID	ISBN	Book-Rating
0	277427	002542730X	1
1	277427	0060930535	0
2	277427	0060934417	0
3	277427	0061009059	1
4	277427	0140067477	0

Figure 4.2: Book-Crossing dataset shape

Data pre-processing

At first, a sampling method is applied to the data frame ending up with only those users who have read at least 40 books and books which at least 20 users have read. Feedback, actually, was explicit as people gave ratings between one and ten. Any registration with rate under five has been deleted because the analysis wasn't interested in what users didn't like. Also, the goal was to mimic smth that is closer to implicit feedback, so the rating values were made binary. Then, 1 represents that user liked the corresponding book and 0 that user hasn't read it yet. Subsequently, with the usage of library "LabelEncoder" from the "sklearn.preprocessing" package, the object-type data transformed to numeric ones. Finally, the imbalance of the data was handled with the "RandomOverSampler" from the "imblearn.oversampling" package, which over-sampled the minority class by randomly picking samples (Figure 4.2).

4.2.2 DBLP-Citation-network-V13 (2021-05-14)

Data presentation

Secondly, for the development of this thesis, experiments were conducted using a part of 10,000 registrations of the DBLP-Citation-network-V13 (2021-05-14) from AMiner.[10] AMiner is a library comprising 5,354,309 papers and 48,227,950 citation relationships. Each record is associated with abstract, authors, year, venue, title, etc. Those citation data were extracted from DBLP, ACM, MAG (Microsoft Academic Graph), and other sources.The schema of the dataset is illustrated in Figure 4.4.

	Authors	Articles	View	Lbl_Authors	Lbl_Articles
0	562d996945cedb3398e78be8	53e9b108b7602d9703b85b88	0.0	640	18
1	53f433efdabfaedce5516b23	53e9b108b7602d9703b85b88	0.0	103	18
2	5628eae445ce1e59660effe6	53e9b108b7602d9703b85b88	0.0	563	18
3	53f38e3edabfae4b34a44275	53e9b108b7602d9703b85b88	0.0	24	18
4	548691a3dabfaed7b5fa2a43	53e9b108b7602d9703b85b88	1.0	497	18

Figure 4.3: Citation dataset schema

Data pre-processing

As previously indicated, each registration of the initial dataframe represents a different paper, and each column contains information about that paper. There are 17 columns comprising information relevant to the abstract of the paper, the authors, the references etc. However, this type of data was inappropriate for the subject of this study. Thus, the necessity of deriving customized implicit feedback to train the RS emerged. Feedback should just give the information that a user has read an article without any further details. Consequently, it was decided that authors would be the users of the RS and reference's papers would be the items they liked. After some modifications the new dataset consists of two columns named "Authors" and "Articles". Following that, the rest of the preprocessing procedure is similar to that of the previous "Book-Crossing" dataset (Figure 4.3).

Field Name	Field Type	Description	Example
id	string	paper ID	43e17f5b20f7dfbc07e8ac6e
title	string	paper title	Data mining: concepts and techniques
authors.name	string	author name	Jiawei Han
author.org	string	author affiliation	Department of Computer Science, University of Illinois at Urbana-Champaign
author.id	string	author ID	53f42f36dabfaedce54dcd0c
venue.id	string	paper venue ID	53e17f5b20f7dfbc07e8ac6e
venue.raw	string	paper venue name	Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial
year	int	published year	2000
keywords	list of strings	keywords	["data mining", "structured data", "world wide web", "social network", "relational data"]
fos.name	string	paper fields of study	Web mining
fos.w	float	fields of study weight	0.659690857
references	list of strings	paper references	["4909282", "16018031", "16159250", "19838944", ...]
n_citation	int	citation number	40829
page_start	string	page start	11
page_end	string	page end	18
doc_type	string	paper type: journal, book title...	book
lang	string	detected language	en
publisher	string	publisher	Elsevier
volume	string	volume	10
issue	string	issue	29
issn	string	issn	0020-7136
isbn	string	isbn	1-55860-489-8
doi	string	doi	10.4114/ia.v10i29.873
pdf	string	pdf URL	//static.aminer.org/upload/pdf/1254/370/239/53e9ab9eb7602d970354a97e.pdf
url	list	external links	["http://dx.doi.org/10.4114/ia.v10i29.873", "http://polar.lsi.uned.es/revista/index.php/ia/ article/view/479"]
abstract	string	abstract	Our ability to generate...
indexed_abstract	dict	indexed abstract	{"IndexLength": 164, "InvertedIndex": {"Our": [0], "ability": [1], "to": [2, 7, ...]}}

Figure 4.4: Data schema (v13) [10]

Chapter 5

The Model

5.1 Matrix Factorization Model (MF)

MF is the most widely used model for RS implementations. MF mathematically reduces the dimensions of the original interaction matrix (user-item matrix) into two smaller sub matrices, the user matrix, and the item matrix. When it comes to prediction, the original matrix is reconstructed by multiplying these two sub matrices. The closer the elements of the new matrix are to 1, the more likely the corresponding user is to interact with the corresponding item. The way the factorization mentioned above is utilized is essential since the loss between the reconstructed and the original matrix should be the least possible. The currently used loss function, called Mean Squared Error (MSE), computes each difference between corresponding elements, squares them and returns their average value [30].

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Where:

- N is the number of elements.
- y_i denotes the observed values.
- q_i are predicted values.

At this point, it is important to note that MF implements a projection of each user and item onto the same latent space of size K. Thus, K denotes the dimension of the latent vectors that represent users and items. Consequently, a way to measure the similarity between two users is to compute their dot product. However, paper [11] argues that inner product limits the expressiveness of latent vectors.

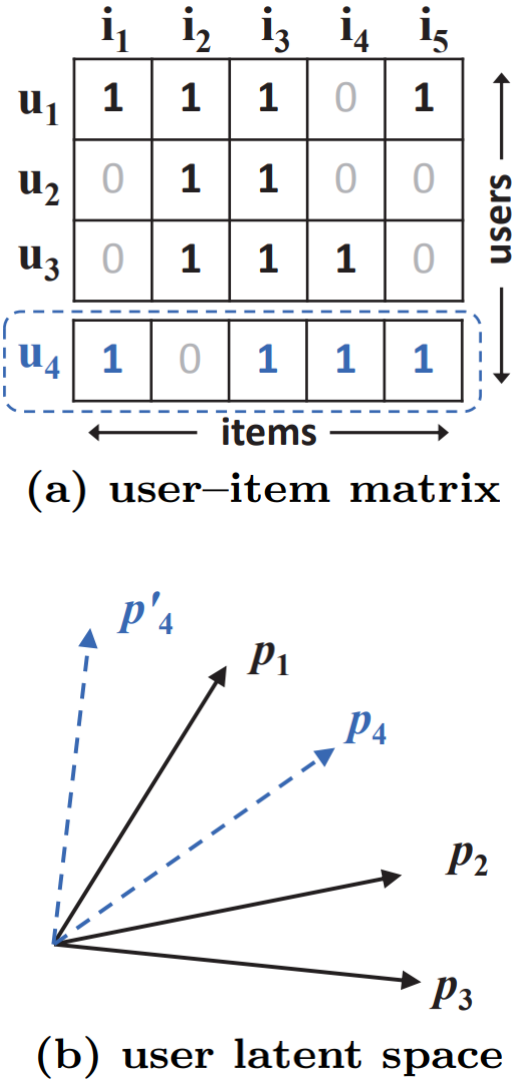

Figure 5.1: An example that illustrates MF's limitation.[11]

Example in Figure 5.1 illustrates that limitation. Computation of the resemblance between users indicates that user u3 is most similar with u2, followed by u1. So, when projecting these users to a 2-dimensional latent space (b) user u2 is placed closer to user u3 than u1. Subsequently, using the same reasoning, user u1 is the most similar with user u4, while user u2 is the least similar. However, regardless of how we place user u4 around user u1, user u3 turns out being the farthest from user u4 even though user u2 is the one which is the most different from user u4. In this manner, it may be concluded that the inner product is incapable of modeling a complex interaction between user and item latent vectors.

5.2 Neural Collaborative Filtering Model (NCF)

To address the issue of MF model's incapability mentioned in the previous section, paper [11] proposes a new NN architecture called Neural Collaborative Filtering (NCF). At first, in the input layer both users and items are one-hot encoded. After that, in the embedding layer a projection onto the latent space takes place. So far, everything seems the same. The main addition to the MF model, is that user and item embeddings are concatenated and then passed through a NN. The NCF layers basically can be any kind of neural connections. For instance, MLP or LSTM can be placed there as shown in Figure 5.2. This way, there are more interesting features such as the nonlinear activation function which is smth that just computing the product between the two embeddings can't offer. Paper [11] claims that with the nonlinearity and the complicated connections in this layer, this model can learn the user and item interactions in a laten space accurately.

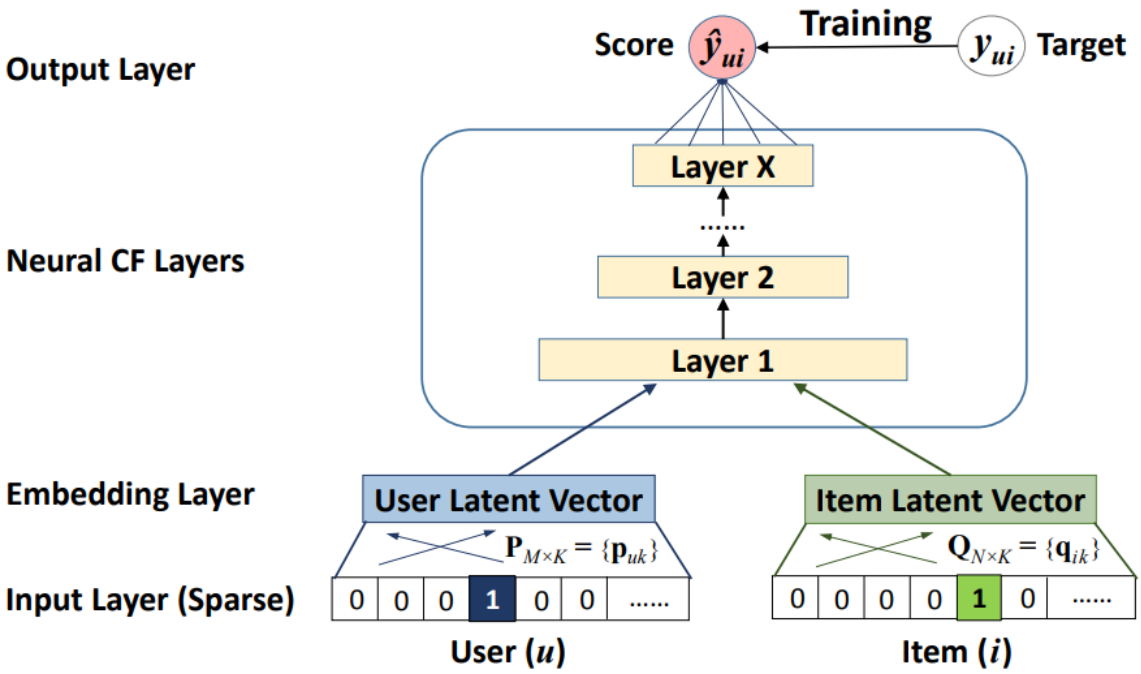

Figure 5.2: Neural collaborative filtering framework.[11]

With that being said, one can easily understand that NCF framework presented above is not much different from the initial method used to learn user and item preferences. More precisely Matrix Factorisation is the most simple case of NCF. The detailed proof, can be found in the Appendix A.

5.3 Neural Matrix Factorization Model

Paper [11] doesn't propose just replacing the MF, but they propose putting it side by side, so that the final model contains two sub modules. Figure 5.3 illustrates that proposal. That way it introduces more nonlinearity while including a NCF module in addition to the original MF layer. In other words, it takes advantage of both the linearity and nonlinearity offered. There are two different embeddings, one for MF layer and also another embedding for the NCF. Then everything is passed through the NeuMF layer, and the two models are fused by concatenating their last hidden layers. Subsequently, this fusion gets through an activation function. In the end, still there is a score and a target, and still the error is computed, expected to be within some boundaries.

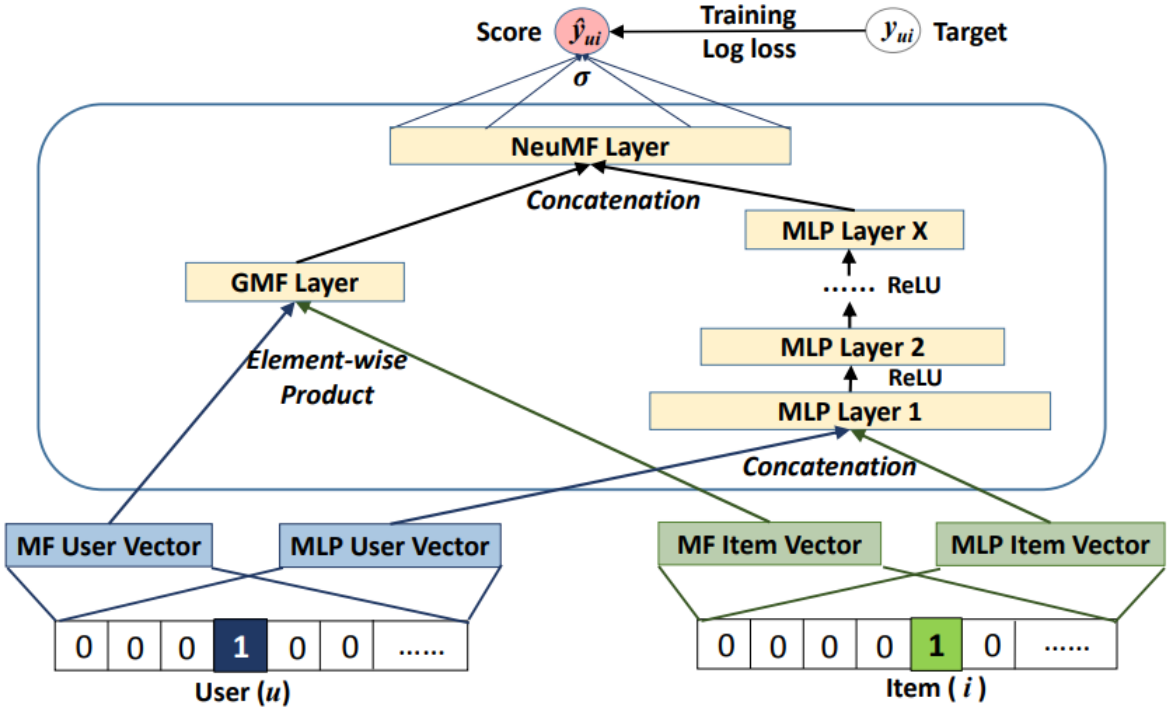

Figure 5.3: Neural matrix factorization model.[11]

Chapter 6

Evaluation Scheme

Training a ML model with suitable data is not enough. The importance of evaluating a model to understand if it needs to be improved or if it is going to perform well with unknown data, is also crucial. Different metrics can be applied to judge a model's performance and are known as evaluation metrics.Evaluation metrics inform if the algorithm is making progress and also evaluate this progress by putting a number on it. This feature of evaluation metrics is an extremely useful feature as it provides the opportunity to compare different models with each other. Moreover, another essential feature of them is their ability to distinguish among model results.

To build a robust model it is critical to choose a metric that fits well to the optimization problem. The kind of suitable evaluation metric is going to differ depending on the type of problem. Even though there is a significant variety of publications about evaluation metrics capable of accomplishing CF, the most popular ones are Precision and Recall. This is due to the fact that their behavior in making recommendations is very efficient, since they are obtaining the most accurate results. One important factor of these metrics is the choice of the number K of predictions to consider when calculating them. More precisely, it refers to the highest K items that the model would be recommending for a user. Lastly, the evaluation of the experiments of this thesis is also accomplished using another widely used metric called accuracy score.

6.1 Confusion Matrix

Confusion matrix is a prerequisite knowledge for understanding the metrics mentioned above and is defined as the table that contributes to performance measurements for some ML tasks of two or more classes output. It includes four different combinations of predicted and actual values, as shown in Figure 6.1. The elements of the confusion matrix can be one of the following:

- True Positive(TP): These are the positive cases that the model rightly predicted.
- False positives(FP): There are the negative cases that the model predicted as positive ones.
- False negatives(FN): There are the positive cases that the model predicted as negative ones.
- True negatives(TN): There are the negative cases that the model rightly predicted.

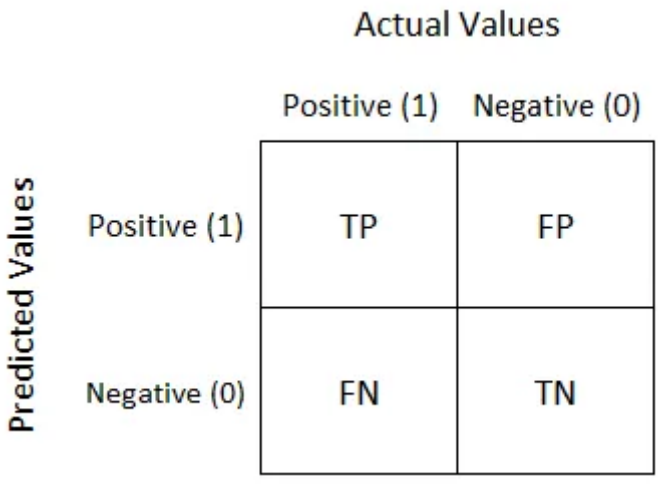

Figure 6.1: Confusion Matrix. [12]

6.2 Accuracy

Accuracy is the most renowned ML model validation method. Generally, accuracy is a metric that describes a model's performance while considering all classes. Simply, it measures how often the model forecasts correctly. Accuracy can be described as the ratio between the number of predictions made correctly and the total number of instances, as shown below.

$$Accuracy = \frac{True_{Positive}+True_{Negative}}{True_{Positive}+TrueNegative+False_{Positive}+False_{Negative}}$$

6.3 Precision

Precision describes how many of the cases that were predicted as correct cases, actually turned out to be positive. It can be defined as the ratio of the number of positive cases which were classified accurately to the total number of cases classified as positive. In other words, precision evaluates a ML model's accuracy in classifying an instance as positive.

$$Precision = \frac{True_{Positive}}{True_{Positive}+False_{Positive}}$$

6.4 Recall(Sensitivity)

Recall explains how many of the actual positive cases the model managed to predict accurately. This metric measures a model's ability to detect positive instances. It can be calculated as the ratio of the number of positive cases that were classified correctly as positive to the total number of positive cases. With that being said, one can easily understand that recall is independent of the negative cases classification.

$$Recall = \frac{True_{Positive}}{True_{Positive}+False_{Negative}}$$

6.5 Precision Vs Recall

Precision versus Recall is an ordinary topic in the field of ML. Most frequently and depending on the problem's requirement, there must be a higher priority on maximizing Precision, or Recall. But when is it wiser to construct a high Precision or a high Recall?

When there is a necessity for making output-sensitive forecasts, the ML models will be in need of high Recall [31]. Predicting critical illness would be an excellent example of this type of problem. Besides, it should be precise and easy to understand that covering any FN cases is essential. Generally, in these kinds of problems, a FP case would have less implications than, for instance, labeling someone who suffers from a serious disease as healthy. Other examples could be relevant to predicting upcoming natural disasters.

On the other hand, a model may have to deal with less critical problems where emphasis is given more on TP and FP cases. On such occasions, Precision offers more insight into the model's skill [31]. A typical example is engines used to detect spam emails. It must be crystal clear that the model doesn't focus on FN cases at all. For instance, it won't have that harmful consequences if a spam email ends up in the inbox folder. Lastly, Recommendation Systems are included in this category, so Precision is more important than Recall for the purposes of this thesis.

Chapter 7

Experiments

7.1 Experimental Settings

7.1.1 MLP Implementation

The MLP network consists of 4 dense hidden layers comprising 64, 32, 16 and lastly 8 fully connected neurons. Neurons have the “ReLu” as their non-linear activation function. Additionally, the model has some Dropout and Batch Normalization layers. All layers are densely connected, which means that each neuron in a previous layer is connected to every neuron in the next layer and so forth. The final layer, which is a fully connected dense layer, performs the operation on the input layers and returns the output. Furthermore, for compilation “Adam Optimizer” is used and the error is calculated by loss function “Binary Cross-entropy”, as it is a recommendation problem.

7.1.2 LSTM Implementation

Similarly to the previous model, the LSTM network has 4 LSTM layers with 32 neurons on each one and 4 Dropout layers densely connected. Each neuron also has the non-linear “ReLu” activation function. Finally, there is an output layer which is a fully connected neuron. The loss function is the “Binary Cross-entropy”, and the optimizer is called “Adam”.

7.2 Hyperparameter Tuning

A ML model can be explained as a mathematical algorithm consisting of several parameters that need to be trained over a set of data. When creating a ML model one can be confronted with a plethora of possible architectures [32]. On most occasions, having the ability to explore those possibilities is more than necessary since knowing the optimal architecture at the beginning is impossible. Parameters which outline a model's architecture and express essential properties of it, are known as hyperparameters. Thus, hyperparameter optimization (or tuning) couldn't be anything different from the procedure of choosing those hyperparameters. Hyperparameter tuning plays a vital role over a model's behavior. If a tuning is not done correctly, the model won't minimize the loss function which means that it will create more errors.

At this point, it is important to distinguish hyperparameters from model's parameters [33]. A ML model estimates or fits its parameters from existing data. Once training is complete, these parameters develop into permanent factors of the model. For instance, when dealing with DNNs each node has a weight value and a bias that show its impact to the final prediction. Those weights and biases are an example of the model's parameters. On the other hand, hyperparameters are variables that control the training process and their estimation is irrelevant with the dataset. Note that changing hyperparameter values leads to production of different parameter values for the same dataset. To conclude, hyperparameter tuning embodies the discovery of an optimal hyperparameter set that can get a model's performance to the maximum.

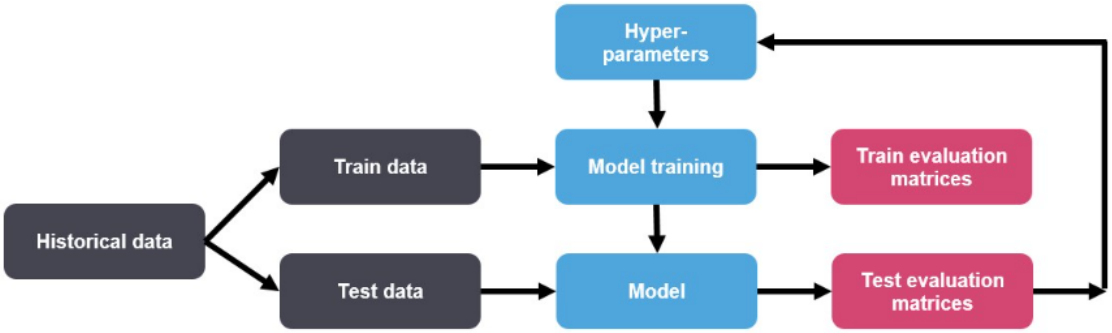

Figure 7.1: Hp Tuning & Evaluation Mechanism [13]

7.3 Performance Comparison

7.3.1 First Experimental Phase

Book-Crossing Experiments

Firstly, using the Book-Crossing dataset, some experiments were executed with different values of the K factor. The key idea behind this factor is as it follows: if a positive interaction of a user with an item is extracted from the dataset, to examine if that item will be in the top K recommendations of the model. No other hyper-parameters were modified during this experimental phase. The observed Precision results for both models are recorded in Table 7.1. There are listed different Precision values for several recommendations between K = 20 and K = 400. The best performance for each model is highlighted. In detail, the top Precision for the first model has been achieved for the case of K = 20 while for the second model for the case of K=100.

Table 7.1: Precision performance for K = 20, 100, 150, 200, 250, 300, 400 recommendations

Algorithm vs K	*20*	*100*	*150*	*200*	*250*	*300*	*400*
NeuMF(MLP)	**0.922**	0.9	0.893	0.85	0.812	0.82	0.77
NeuMF(LSTM)	0.87	**0.941**	0.88	0.86	0.864	0.85	0.847

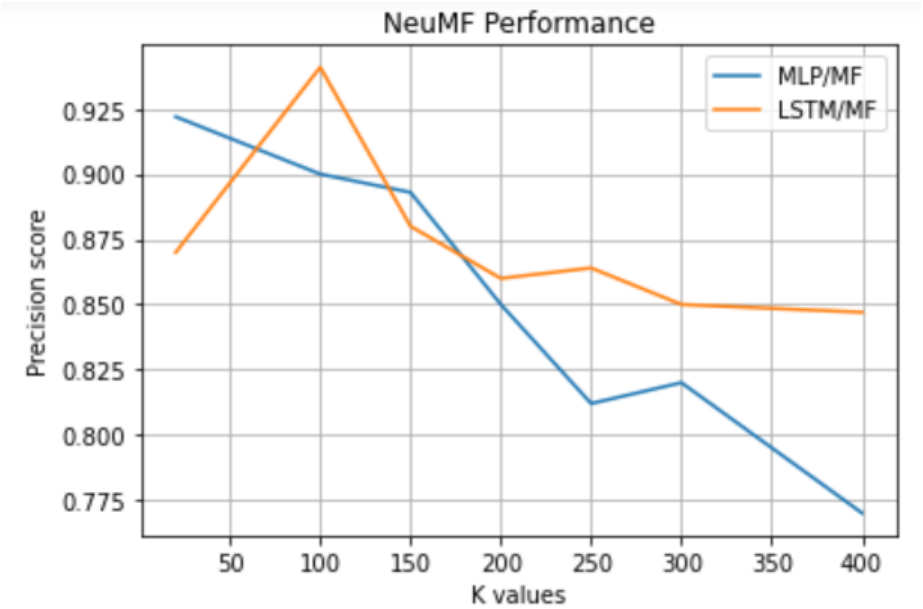

Figure 7.2: Precision performance for K recommendations

AMiner Citation Network Experiments

The same experiments for the two versions of the NeuMF, were executed with the DBLP-Citation-network-V13 dataset which recommends publishing venues. The Precision performance for the same alternatives of K values is presented in Table 7.2. The first model accomplishes an excellent score when making 100 recommendations, while the second model works best with K =20.

Table 7.2: Precision performance for K = 20, 100, 150, 200, 250, 300, 400 recommendations

Algorithm vs K	*20*	*100*	*150*	*200*	*250*	*300*	*400*
NeuMF(MLP)	0.98	**0.99**	0.986	0.986	0.985	0.98	0.976
NeuMF(LSTM)	**0.99**	0.97	0.933	0.94	0.936	0.923	0.925

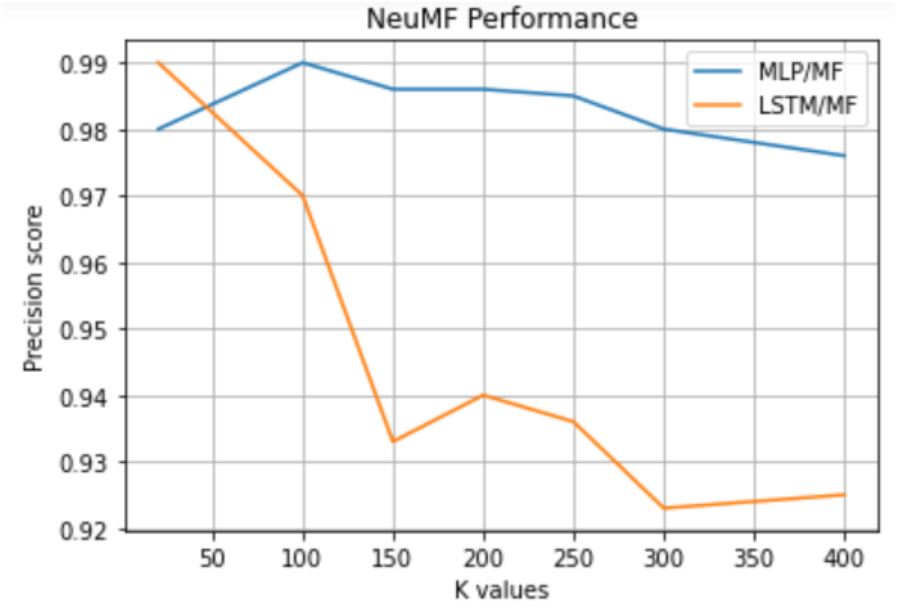

Figure 7.3: Precision performance for K recommendations

7.3.2 Second Experimental Phase

Book-Crossing Experiments

Moreover, the two models were fitted on different numbers of epochs (the amount of times data is passed through the network) in the range [10, 300]. During these tests, the K factor for each model was set to the one provided their best score in the previous experimental phase. Tables 7.3 and 7.4 illustrate the results gathered from this procedure.

Table 7.3: Precision performance for the tested activations at K = 20 and K = 100

Algorithm vs Epochs	*10*	*40*	*80*	*100*	*150*	*300*
NeuMF(MLP)	0.882	**0.93**	0.881	0.764	0.647	0.588
NeuMF(LSTM)	0.85	0.9	**0.941**	0.8	0.76	0.81

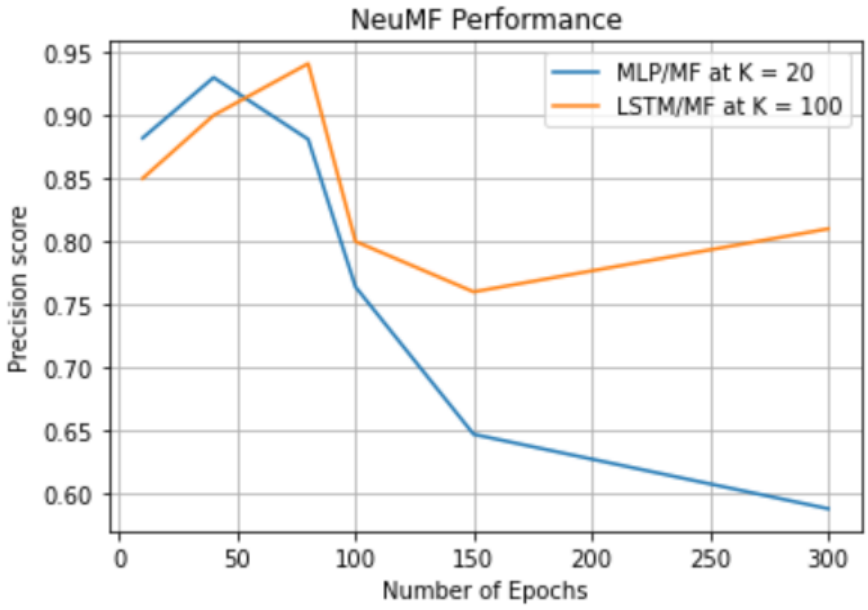

Figure 7.4: Precision performance for different number of epochs

Table 7.3 shows Precision performance while Table 7.4 Accuracy score. At this point, it can be noted that NeuMF utilizing MLP has recorded its best performance at only 40 epochs of training. On the other hand, NeuMF utilizing LSTM achieves the highest Precision score after 80 epochs. In addition, Table 7.4 describes that both models perform best, according to the Accuracy metric, for a number of epochs in the range [80,150].

Table 7.4: Accuracy score

Algorithm vs Epochs	*10*	*40*	*80*	*100*	*150*	*300*
NeuMF(MLP)	0.732	0.766	0.801	0.827	**0.843**	0.841
NeuMF(LSTM)	0.725	0.794	0.831	0.838	**0.858**	0.823

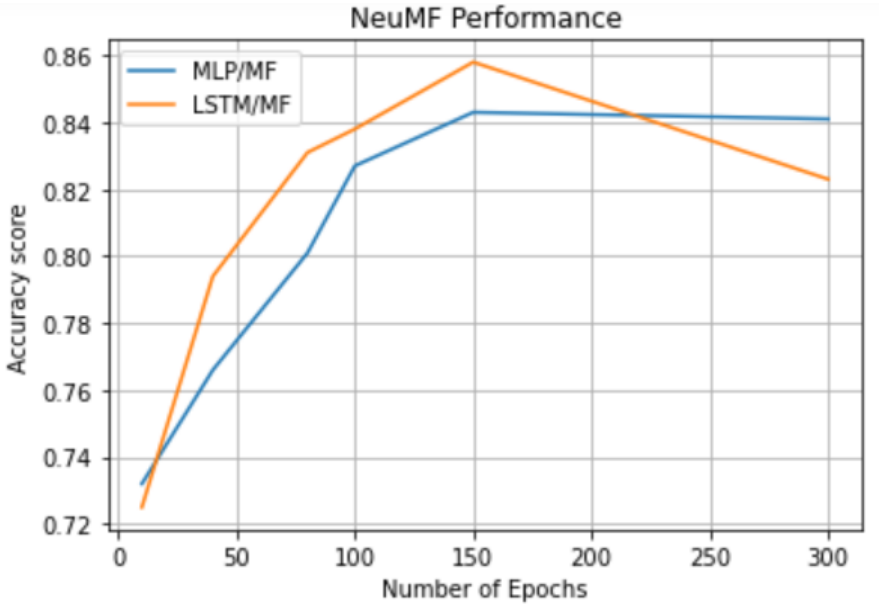

Figure 7.5: Accuracy score for different number of epochs

AMiner Citation Network Experiments

Table 7.5 shows the evolution of the Precision metric for each of the epoch numbers used in the study. One can easily notice that both networks give exceptional results. At this point, it should be mentioned that the previous process was carried out for each of the following epochs: 10, 40, 80, 100, 150 and 300 and one would expect that more iterations of the process would yield better results. Nevertheless, in ML having a huge number of epochs leads to overfitting. In other words, the model absorbs the noise of the training data to an extent that affects its performance with new data in a negative way. To specify, the model does not have a satisfactory ability to generalize. Lastly, similar conclusions can be extracted from Table 7.6, which gives the corresponding Accuracy results.

Table 7.5: Precision performance for the tested activations at K = 100 and K = 20

Algorithm vs Epochs	*10*	*40*	*80*	*100*	*150*	*300*
NeuMF(MLP)	0.94	0.94	0.96	**0.98**	0.97	0.971
NeuMF(LSTM)	0.972	0.979	**0.99**	**0.99**	0.984	0.98

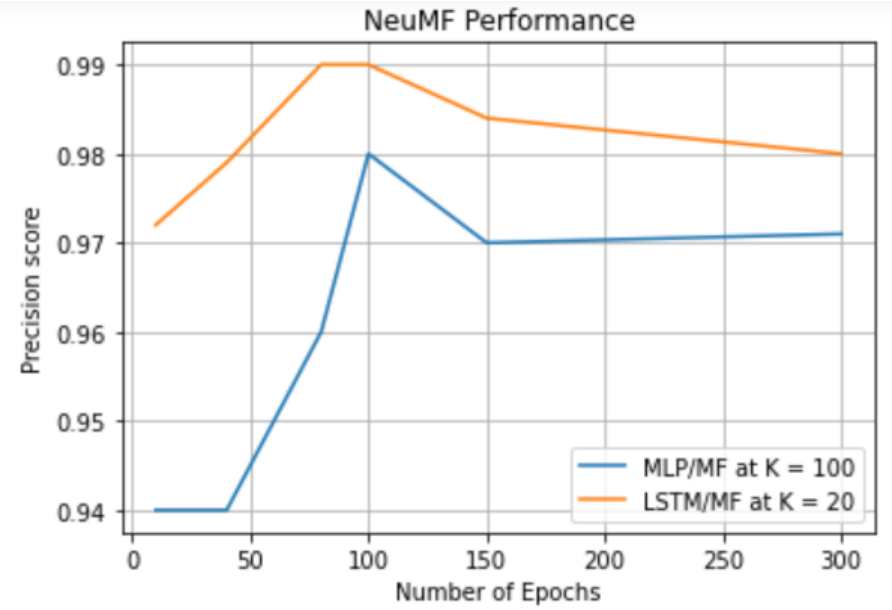

Figure 7.6: Precision performance for different number of epochs

Table 7.6: Accuracy score

Algorithm vs Epochs	*10*	*40*	*80*	*100*	*150*	*300*
NeuMF(MLP)	0.798	0.851	0.878	**0.909**	0.866	0.862
NeuMF(LSTM)	0.773	0.85	**0.914**	0.89	0.877	0.872

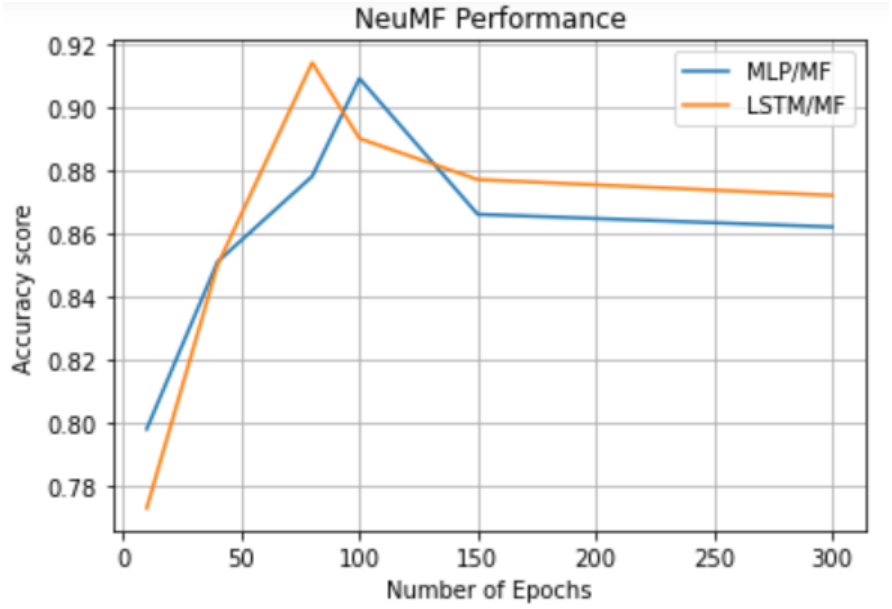

Figure 7.7: Accuracy score for different number of epochs

Chapter 8

Tools

There is a plethora of alternative tools one can utilize for the development of recommendation systems like NeuMF presented above. However, for the aims of this thesis the following language and tools were chosen.

8.1 Development Environment

Anacoda is an environment which provides Python and R language distribution in a convenient way [34]. Nowadays, more than 20 million people use this technology for data science tasks implementation. Anaconda offers thousands of open-source packages and libraries. Apart from that, it allows fast innovating without sacrificing security. Lastly, the following apps and tools are available via Anaconda's user-friendly interface, while Jupyter Notebook is the code editor selected for this thesis implementation.

- Visual Studio Code
- RStudio
- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glue
- Orange

8.2 Programming Language

Python is the programming language selected for developing this work. To be more precise, the exact version used is 3.8.5. Python is an object-oriented, high-level programming language generated from Guido van Rossum [35]. It has a incredibly easy and comprehensible syntax, succeeding this way notable maintenance cost reduction. In addition, an extensive number of libraries are accessible to all prime platforms freely. Furthermore, the debug procedure is really fast since no compilation is taking place. For these reasons, from 1991 until today Python is one of the most widely used programming languages. To conclude, Python may appear to be useful for many purposes such as those listed below.

- Application development
- Complex mathematical calculations
- Script production for existing modules connection
- Link to DB systems

Libraries

Different libraries were employed to improve computational complexity of the algorithms:

- **pandas** version 1.1.3 is used to analyze data [36]
- **numpy** version 1.19.2 is used for working with arrays and scientific computing in general [37]
- **sklearn** version 1.1.2 ([38]) is used for following packages:
    1. **preprocessing.LabelEncoder** for encoding
    2. **model-selection** for splitting dataset to train and test
    3. **metrics** for importing accuracy score metric
- **imblearn** version 0.9.1 is used for providing RandomOverSampler technique in order to avoid unbalance of the data[39]
- **tensorflow** version 2.7.0 ([40]) is used for following packages:
    1. **keras.layers** for importing different layers of the NN architectures such as Concatenate, Dense, Embedding, Flatten, Input, Multiply, LSTM, Dropout, Reshape, BatchNormalization layers
    2. **keras.models** for importing module Model

3. **keras.optimizers** for importing Adam optimizer
4. **keras.metrics** for importing Precision and Recall metrics

8.3 Computer Specifications

The code has been tested on DELL-INSPIRON -5559 GL -PROCESSOR CORE I7 -6500U -2.50GHZ -8GB RAM -1 TB HDD -VGA 4GB RADEON -DOS -15.6 INCH.

Chapter 9

Conclusions

9.1 Concluding Remarks

In this work, we have demonstrated that state-of-the-art performance can be achieved by jointly performing deep representation learning and matrix factorization for recommendation. Neural matrix factorization models are implemented to bridge the gap between deep learning models and collaborative filtering. The main focus of this thesis is to survey the NeuMF framework presented in [11] and develop two different instantiations of it. Experimental results illustrate that both approaches (MLP and LSTM) have almost identical accuracy but with some minor differences. Furthermore, we have conducted a comparison of different HP like factor K and training epochs for two real-world datasets. An appropriate number K of items that we've recommended plays a vital role in the model's performance while a proper number of training epochs is crucial for avoiding under-fitting and over-fitting. Regarding the computational cost, both NeuMF instantiations have slower implementation than MF because of compilation encumbrance and their different components that must be defined such as the functions. However, it is certainly an improvement in CF, since it includes the MF technique and it should be at least as efficient as it. Flexibility is also a benefit as neural network architectures allow testing of different numbers of layers, activation functions etc. in order to implement something more tailor made for each dataset.

9.2 Future Work

When it comes to the implementation of CF systems, there are a couple of hardships one comes up against with. The first one is relevant with the type of feedback sb gets, as they are explained in Section 4.1. In detail, the problem of how to deal with the missing values in the very sparse matrix

of data arises when this data is given in an implicit manner. For explicit data the missing values must be treated as unknown fields that the RS is going to predict. On the other hand, for implicit data a missing value can express two different situations. It either means that the user didn't like the corresponding item, or that the user was unaware of it. In that sense, it is slighlty harder to evaluate the RS. Furthermore, another issue with CF is the so-called "cold start problem". When a user is signing up to a recommendation platform there isn't any past information to rely on in order to locate similar users. To overcome these two issues CBF becomes necessary. For those reasons most RS are hybrid systems between the two techniques stated in Section 4.1. Hence, among the possible extensions that could be made to this thesis approach is endorsing a hybrid alternative. Moreover, regarding future work, the experimental phase could be expanded while enclosing other hyper-parameters. Those HP can be different activation functions, number of hidden layers and neurons in each layer etc.

9.3 Code Repository

This thesis's code is hosted on Github in the following repository:

`https://github.com/VasiaKara/COLLABORATIVE-FILTERING-UTILIZING-NEURAL-NETWORKS`

Appendix B is a guide on how to download every required tool in order to run the project correctly.

Bibliography

[1] Anuja Arora Anu Taneja. Recommendation research trends: review, approaches and open issues. 2018.

[2] Arne Wolfewicz. Deep learning vs. machine learning – what's the difference? November 2002.

[3] Adel Mohsenzadeh Ali Haghighat mesbahi Jalal Foroozesh, Abbas Khosravani. Application of artificial intelligence (ai) modeling in kinetics of methane hydrate growth. 2013.

[4] Hector Rene Vega-Carrillo Jose Manuel Ortiz-Rodriguez, Ma. del Rosario Marti?nez-Blanco. Evolutionary artificial neural networks in neutron spectrometry. 2011.

[5] Aravindpai Pai. Cnn vs. rnn vs. ann – analyzing 3 types of neural networks in deep learning. https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/.

[6] Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation. 2018.

[7] Mehmet Toprak. Collaborative filtering. 2020.

[8] Google Developers. Collaborative filtering. https://developers.google.com/machine-learning/recommendation/collaborative/basics.

[9] Data preprocessing in data mining, Last Updated : 29 Jun, 2021. https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/.

[10] Citation network dataset-citation-network v13, 2021. https://www.aminer.org/citation.

[11] Hanwang Zhang Liqiang Nie-Xia Hu Xiangnan He, Lizi Liao and Tat-Seng Chua. Neural collaborative filtering. 2017.

[12] Sarang Narkhede. Understanding confusion matrix. 2018.

[13] Arne Wolfewicz. Analytics vidhya. April 2021.

[14] Steeve Huang. Neural collaborative filtering (ncf) explanation implementation in pytorch, 2018. https://www.youtube.com/watch?v=O4lk9Lw7lS0.

[15] Robin Burke. Hybrid recommender systems: Survey and experiments. November 2002.

[16] Jon Herlocker Shilad Sen J. Ben Schafer, Dan Frankowski. Collaborative filtering recommender systems.

[17] https://movielens.org/.

[18] Debajyoti Mukhopadhyay Dheeraj kumar Bokde, Sheetal Girase. Role of matrix factorization model in collaborative filtering algorithm: A survey.

[19] Bin Cao Nathan N. Liu Rajan Lukose Martin Scholz Rong Pan, Yunhong Zhou and Qiang Yang. One-class collaborative filtering. 2008.

[20] Yehuda Koren Yifan Hu and Chris Volinsky. Collaborative filtering for implicit feedback datasets. 2008.

[21] Naiyan Wang Hao Wang and Dit-Yan Yeung. Collaborative deep learning for recommender systems. 2015.

[22] Eleni Tousidou Vaios Stergiopoulos, Michael Vassilakopoulos and Antonio Corral. Hyper-parameters tuning of artificial neural networks: An application in the field of recommender systems.

[23] Yoshua Bengio Yann LeCun and Geoffrey Hinton. Deep learning. 2015.

[24] Carolina Bento. Multilayer perceptron explained with a real-life example and python code: Sentiment analysis. 2021.

[25] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding lstm - a tutorial into long short-term memory recurrent neural networks. 2019.

[26] M.-Y. Kan X. He, H. Zhang and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. *SIGIR*, page 549–558, 2016.

[27] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. *KDD*, page 426–434, 2008.

[28] Fanny Chevalier Ravin Balakrishnan Jian Zhao, Christopher Collins. Interactive exploration of implicit and explicit relations in faceted datasets. *IEEE*, 2013.

[29] Cai-Nicolas Ziegler. Book-crossing dataset. http://www2.informatik.uni-freiburg.de/~cziegler/BX/.

[30] Mean squared error. https://en.wikipedia.org/wiki/Mean_squared_error.

[31] Arjun Kashyap. Precision and recall — a simplified view. 2019.

[32] JEREMY JORDAN. Hyperparameter tuning for machine learning models. 2017.

[33] Juan Navas. What is hyperparameter tuning? 2022.

[34] Anaconda: the world's most popular data science platform, 2023. https://www.anaconda.com/products/practitioner-tools.

[35] Python.org. https://www.python.org/doc/essays/blurb/.

[36] Pandas documentation. https://pandas.pydata.org/docs/.

[37] Numpy documentation. https://numpy.org/.

[38] Scikit documentation. https://scikit-learn.org/stable/.

[39] Imbalanced-learn documentation. https://imbalanced-learn.org/stable/.

[40] Tensorflow documentation. https://www.tensorflow.org/learn.

[41] Anaconda installation. https://docs.anaconda.com/anaconda/install/.

APPENDICES

Appendix A

MF as NCF Instance

A question that a person might wonder is how can MF be a particular case of the NCF framework. The following analysis is a proof of this claim. Firstly, consider replacing all the layers of the NCF with just a multiplication layer, as Figure A.1 depicts. A multiplication layer multiplies the inputs it gets element-wisely. In addition, the coefficients used from the NCF layers to the output layer are equated to a unit matrix. The term unit matrix is used for a scalar matrix whose elements are all equal to one. Moreover, as a transfer function the linear function is used.

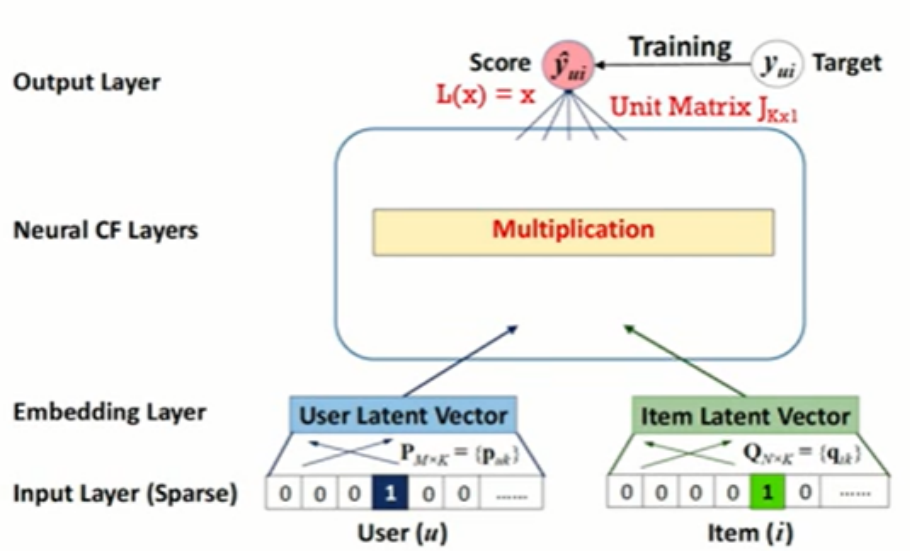

Figure A.1: MF as a NCF instance [14].

With that been said, we can conclude to the following equations:

$$\hat{y}_{ui} = L(p_u \odot q_i \times J_{K\times 1}) \Rightarrow$$

Since weights are a unit matrix, y is assigned to each of the entries in the matrix that it is being multiplied on. So, becomes the dot product between p_u^T and q_i.

$$\hat{y}_{ui} = L(p_u^T \cdot q_i) \Rightarrow$$

Lastly, L is a linear activation function. thereafter the input equals the output, so we have this final form d this is the exact same form of MF.

$$\hat{y}_{ui} = p_u^T \cdot q_i$$

Where:

- $\hat{y}_{ui}$ denotes the predicted value of the unobserved entries for user u and item i.
- p_u and q_i are the laten vectors of user u and item i.
- $\odot$ is the sign depicting the element-wise product.

Appendix B

Prerequisites

Appendix B is a guide on how to download every required tool in order to run the project correctly.

- At first, you need to install the Anaconda environment. Review the system requirements listed in [41] and if necessary, install Miniconda which simply is a mini version of Anaconda. Additionally, download the most recent Python 3 release.
- After the installation of Anaconda is completed, you can open Anaconda Prompt. For instance, if you are a windows user, you will have to search for it at the Windows start menu as shown in Figure B.1. However, it is also available for MacOS, Linux/UNIX and other systems.

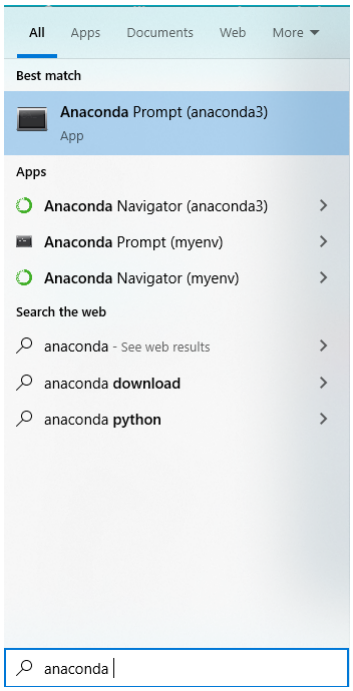

Figure B.1: Anaconda Prompt search

- To make sure that everything is installed successfully type python in the prompt and hit Enter. You should get something like Python 3.5.8.
- Next, after navigating to the project's file, you have to install the requisite dependencies for it, which are placed in a .txt file. In detail, you will do this through pip3 using the following command in anaconda prompt:
  pip3 install -r requirements.txt
- Finally, in anaconda prompt type jupyter notebook and hit Enter. This way Jupyter notebook home page will appear on your screen (Figure B.2). Then, all you have to do is find the .ipynd files of this project.

Figure B.2: Jupyter Notebook home page