PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF UNIVERSITY - M'SILA

FACULTY OF MATHEMATICS
AND INFORMATICS
DEPARTMENT OF
COMPUTER SCIENCE
N°……

DOMAINE: MATHEMATICS
AND INFORMATICS
FILIERE: COMPUTER SCIENCE
OPTION: INFORMATION SYSTEMS
AND SOFTWARE ENGINEERING

A dissertation submitted in fulfillment of the requirements
For the Degree of master

By Akram AZOUZI & Ameur BEN MIMOUNA

Subject:

Deep learning based recommender systems

Presented to the jury :

DEBBI Aimad-Eddin	University of M'sila	President
BOUZAROURA Ahlem	University of M'sila	Reporter
YAGOUBI Rached	University of M'sila	Examiner

Academic Year: 2021/2022.

Dedication

To everyone that supported, helped
or motivated me in this life, for
everyone that believed in me every time, for
my parents, friends and family.

Akram AZOUZI

My parents “There are not enough words
To tell you how grateful I am to you. Thank you for everything
You did it for me. I would also like to thank my big family and friends for being with me all the time

Ameur BEN MIMOUNA

Acknowledgements

We would like to express our thanks to supervisor Mrs.BOUZAROURA Ahlem who supported and followed up this work. Also great thanks to the jury president and jury members for agreeing to judge this work. We would like to thank the managers and all the staff of the computer science Department of M'sila for the facilities they have granted us to complete this work, Not forgetting all the professors who guided us well throughout our curriculum .

Table of contents

List of figures and tables

Chapter 01 Deep Learning

Chapter 02 Recommender systems

Chapter 03 Deep learning techniques in recommender systems

Chapter 04 Implementation and Experimental Results

List of equations

General introduction

Perhaps you have heard of the recommender systems or recommendations someday before. If not, you have been exposed to them directly or indirectly, how?! Have you observed that most of what you buy or consume was chosen through some kind of recommendation. maybe from friends, family, society or from sources selling you the product or service. When you browse the Internet and open youtube, for example, you will see a list of videos and auto generated playlists that perhaps you will like based on your past watching and history. Instagram suggests people you may know and would probably like to follow. It also curates a News Feed for you based on the posts you've liked or seen, the people you've be-followed and the profiles you've checked before. Aliexpress recommends items to you as you browse for a particular product. It shows you similar products from a competing source and suggests auxiliary items frequently bought together with the product.

So, a good recommendation is the core of successful business for these companies. It's in the best interests of these companies to engage you with content that you love so that you continue to subscribe to its service, the more relevant the items Amazon shows you, the greater your chances – and volume – of purchases will be, which directly translates to greater profits. Equally, establishing friendship is key to Facebook's power and influence as an almost invincible social network, which it then uses to churn money out of advertising.

Our work fits into this context, how to design a recommendation system based on deep learning techniques?

So, the objective of our work is the design and the realization of a Deep Learning based recommender system capable of recommending items according to user preferences and interests.

Manuscript Organization

This document is organized into the following chapters:

Chapter 1: Deep learning.

Chapter 2: Recommender systems.

Chapter 3: Deep learning techniques in recommender systems.

Chapter 4: Implementation and experimental results.

Chapter 01

Deep Learning

Introduction

Artificial intelligence (AI) is the scientific discipline of creating computer programs that perform operations comparable to those of the human mind, such as learning or logical reasoning. Machine learning (ML) goes further, it is this branch of AI that allows machines to learn by themselves, without relying on commands. It refers to the development, analysis and implementation of methods that allow a machine to evolve and perform tasks impossible for a human being through a learning process. Deep learning (DL) is a type of machine learning, but more complex. It is a set of ML methods attempting to model with a high level of data abstraction through articulated architectures of different nonlinear transformations. In this chapter we will first present the basic notions of machine learning and define deep learning, then we will present neural networks and their main architectures.

1. Machine learning

According to Tom Mitchell**[1]** :"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E". Machine learning is an area of artificial intelligence concerned with the development of techniques that allow computers to learn. Learning is the ability of the machine to improve its performance based on previous results**[2]**.

1.1 Types of Machine learning

Machine learning is subdivided according to the nature of the input data and then according to the nature of the output data.

1.1.1 Machine learning according to the nature of the input data

- Supervised learning: This is a form of machine learning that creates artificial intelligence models based on "tagged" learning data.

- Unsupervised learning: It's a machine learning technique that looks for a structure of unlabeled data.
- Reinforcement learning: the algorithm learns a behavior from an observation. The algorithm's action on the environment produces a return value guiding the learning algorithm.

1.1.2 Machine Learning according to the nature of the output data

1) **Classification**: classification is a type of supervised learning, which identifies the category to which a new element belongs on the basis of a data set of data training containing observations of which the category is known**[3]**. There are several types of classification algorithms:
    - Decision Tree: Classifies the use of a tree structure with if-then rules, executing the input through a series of decisions until it reaches a termination condition. Able to model complex and highly intuitive decision processes, but can easily oversize data – Random forest: a set of decision trees, with automatic selection of the most efficient tree. Provides the strength of the decision tree algorithm without over-learning problems**[4]**.
    - Naïve Bayes classifier: a probability-based classifier. Calculates the probability that each data point exists in each of the target categories. Simple to implement and precise for a wide range of problems, but sensitive to all the categories selected.**[5]**.
    - K-Nearest Neighbor: Classifies each data point by analyzing its closest neighbors among learning examples**[6]**. Simple to implement and understand, effective for many problems, especially those with low dimension. Provides lower accuracy than the supervised algorithms, and requires a lot of calculations.
2) **The Regression**: regression is a supervised learning technique, it is the pre- diction of continuous numerical values for a set of data from a learning base, it predicts a continuous output variable (y) based on the value of one or more predictor variables (x). The inputs are called independent values, the output is called the dependent value. There are weights called coefficients, which deter- mine how much each input value contributes to the result, or its importance **[7]**. There are many types of regression :

- Linear regression: suitable for dependent values that can be adjusted with a straight line (linear function).
- Polynomial regression: suitable for dependent variables that can be adjusted by a curve or series of curves.
- Logistic regression: adapted to dependent variables that are binary and therefore not normally distributed.

2. Deep Learning

The idea of Deep Learning is not a recent one, but it actually dates back to the 1980s, more particularly following the work of multilayer neural networks and the work of some pioneers of Machine Learning and Deep Learning such as the French Yann LeCun**[8]**. Together with two other computer scientists, Kunihiko Fukushima and Geoffrey Hinton, they developed a particular type of algorithm called the Convolutional neural network (CNN)**[9]**.

Although this approach gives its results, its progress and evolution is limited by technological advances in microprocessors, computational power, and lack of access to data to train neural networks. However, some researchers have continued to work on this model for about two decades and with the help of technological developments, but especially with the ever-increasing availability of data, have been able to improve this technique.

In 2007 the Stanford Vision Lab, with Fei-Fei Li at its head, developed an aggregator of images where several million ImageNet photos are recorded and tagged. In 2010, ImageNet included 15 million images all categorized according to their own characteristics (vehicles, animals, etc.).

In 2012 Deep Learning was brought up to date with a resounding success at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which is an annual image recognition competition founded by Stanford University, as part of its STANFORD VISION LAB. Several teams of computer scientists compete in this competition every year to award the victory to the program with the lowest failure rate. And while deep learning algorithms are absent from the competition, in 2012 it is indeed a Deep Learning algorithm that will win the 2012 edition to the general surprise**[10]**.

2.1 Definition

Deep learning is a process that allows computers to learn to execute activities that are inherent in the brain, such as picture identification, as a new branch of machine learning research.

Today, the deep learning approach DL is the new trend in machine learning, as it provides far more advanced pattern recognition and picture categorization than the old machine learning approach ML **[11].** Figure.I.1 shows an illustrative diagram of deep learning with multiple layers.

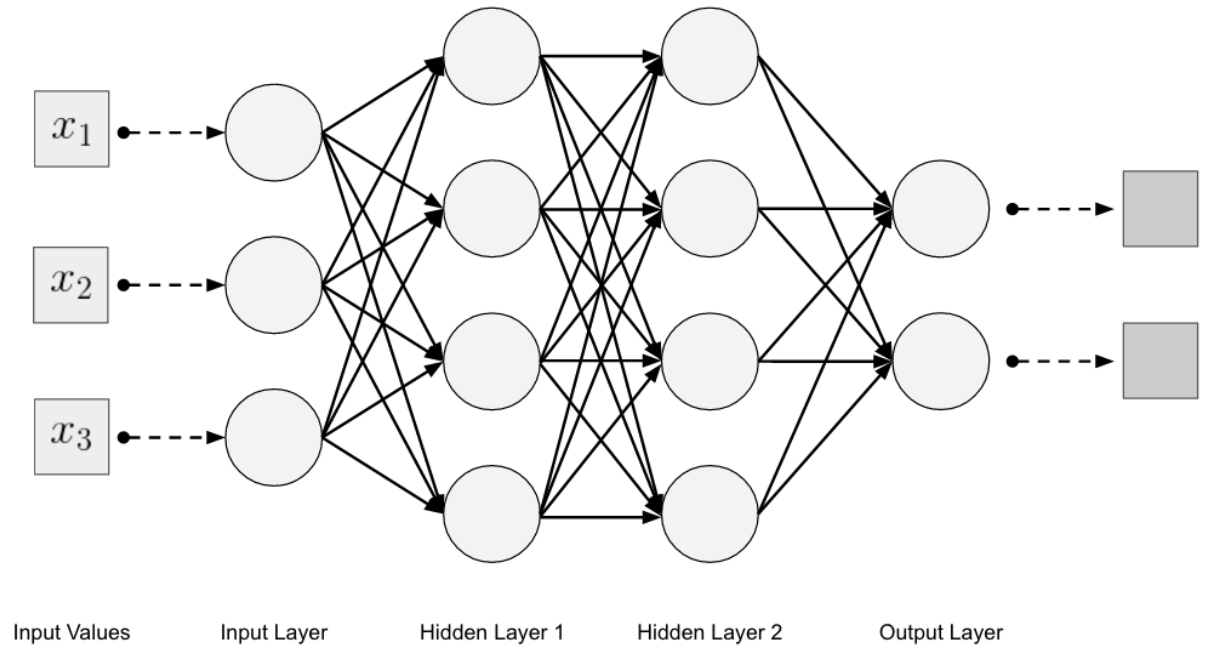

Figure 1.1: multilayer neural network architecture[7].

2.2 Advantages of Deep Learning

Deep learning is a true revolution in AI and goes well beyond machine learning. It is able to solve a wide variety of AI problems, such as:

- Improve traditional development in AI tasks.
- Exploit a large amount of data such as big data.
- Adapt to several types of problems.
- Extract characteristics automatically.

Figure 1.2 below explains the difference between machine and deep learning. Machine learning depends on a human-generated dataset that teaches the algorithm how to define data. On the other hand, deep learning, when a neural network processes an input, it creates its own layers depending on the input and output of the data. To recap the difference between machine learning and deep learning:

- Machine learning uses algorithms to analyze data, learn from it and make informed decisions based on what they have learned.
- Deep learning structures layered algorithms to create an artificial neural network (ANN) capable of learning and making intelligent decisions on its own.
- Deep learning is a subfield of machine learning. Although both belong to the broad category of artificial intelligence, deep learning is what powers the most human-like artificial intelligence.

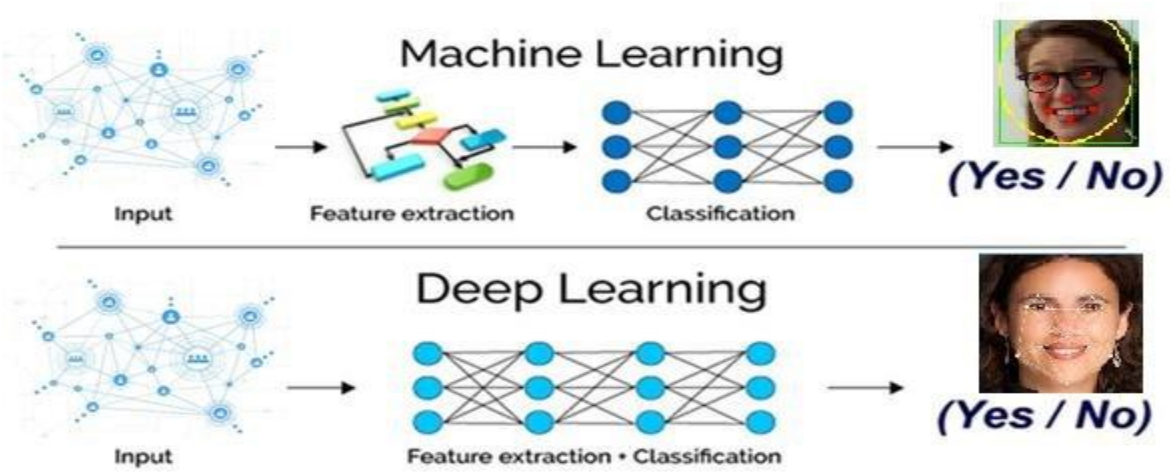

Figure 1.2: Difference between Machine Learning and Deep Learning[8].

2.3 Application domains of deep learning

Learning models are developing in the field of computer science applied in robotics, bioinformatics , pattern recognition , cyber security, healthcare, Computer-aided pedagogy and more generally artificial intelligence. Deep learning can, for example, allow a computer to better recognize highly deformable objects and/or analyze for example the emotions revealed by a photograph or filmed face, or analyze the movements and position of the fingers of a hand,

which can be useful for translating sign languages, improving the automatic positioning of a camera, etc. They are used for certain forms of medical diagnostic assistance (e.g.: automatic recognition of cancer in medical imaging), or prospective or prediction (e.g., predicting the properties of a soil filmed by a robot).

3. Neural networks

Neural networks were born in 1943 thanks to W.MCCulloch and W.Pitts **[12]**, they were able to demonstrate that the brain is equivalent to a Turing machine, thought then becomes purely material and logical mechanisms. They stated in 1955 ”The more we learn about organisms, the more we are led to conclude that they are not merely analogous to machines, but what about this machine. McCulloch and Pitts’ demonstration was one of the important players in the creation of cybernetics. In 1949, D. Hebb **[13]** presented in his book “The Organization of Behavior” a learning rule. Many network models today are still inspired by Hebb’s rule. In 1958, F.Rosenblatt [10] developed the Perceptron model. It is a neural network inspired by the visual system. It has two layers of neurons, a perception layer and a decision-making layer. It is the first artificial system capable of learning by experience. In the same period, the ADALINE model (ADAptive LINear Element) was presented by B.Window **[14]**, an American researcher at Stanford. This model will subsequently be the basic model of multilayer networks. In 1969, M. Minsky and S.Papert published a review of the properties of the Perceptron. This is going to have a big impact on research in this area. It will decrease sharply until 1972, when T.Kohonen **[15]** presented his work on associative memories and proposed applications to pattern recognition. It was in 1982 that J.Hopfield presented his study of a completely looped network, of which he analyzed the dynamics **[16]**.

3.1 Definition

A neural network is the association, in a more or less complex graph, of elementary objects, the formal neurons. The main networks are distinguished by the organization of the graph, that is to say their architecture, its level of complexity (the number of neurons, presence or not of feedback loops in the network), by the type of neurons (their functions of transition or activation) and finally, the goal: supervised or unsupervised learning, optimization, dynamic systems, etc**[17]**.

3.2 Artificial and deep neural networks

Artificial neural networks are a supervised learning system made up of a large number of simple elements, called neurons or perceptrons. Each neuron can make simple decisions and feed its decisions to other neurons, organized in interconnected layers. Together, the neural network can emulate almost every function and answer virtually every question, with enough samples of training and computing power. A shallow neural network consists of only three layers of neurons as shown in Figure 1.3.(B)

- An input layer that accepts model-independent variables or inputs.
- A hidden layer.
- An output layer that generates predictions.

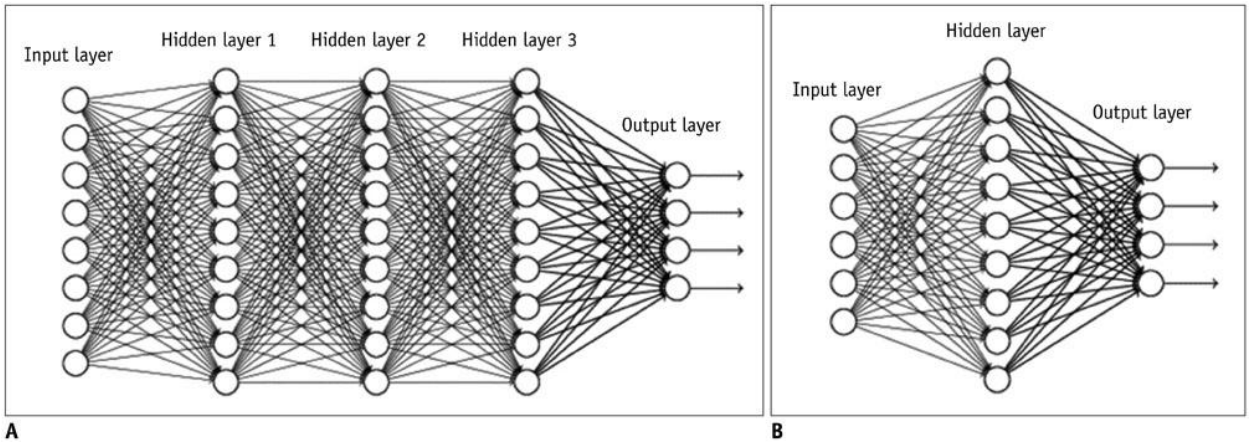

Figure 1.3: the difference between deep and shallow neural networks **[18]**.

A deep neural network has a similar structure, but it has two or more ”hidden layers” of neurons that process inputs (see Figure 1.3.(A)). Goodfellow, Bengio and Courville **[19]** have shown that while shallow neural networks are capable of solving complex problems, deep neural networks are more accurate and improve accuracy as new layers of neurons are added. Additional layers are useful up to a limit of 9 to 10. Today, most neural network models and implementations use a deep network between 3 and 10 layers of neurons.

3.3 Activation functions

An activation function is a mathematical equation that determines the output of each element (perceptron or neuron) in the neural network. As shown in Fig 1.4, it takes the input of each neuron and transforms it into an output, usually between a 0 or -1 and 1. Classic activation functions used in neural networks include the sigmoid function and tanh. New activation functions, intended to improve the efficiency of neural networks, include ReLu and Swish **[20]**.

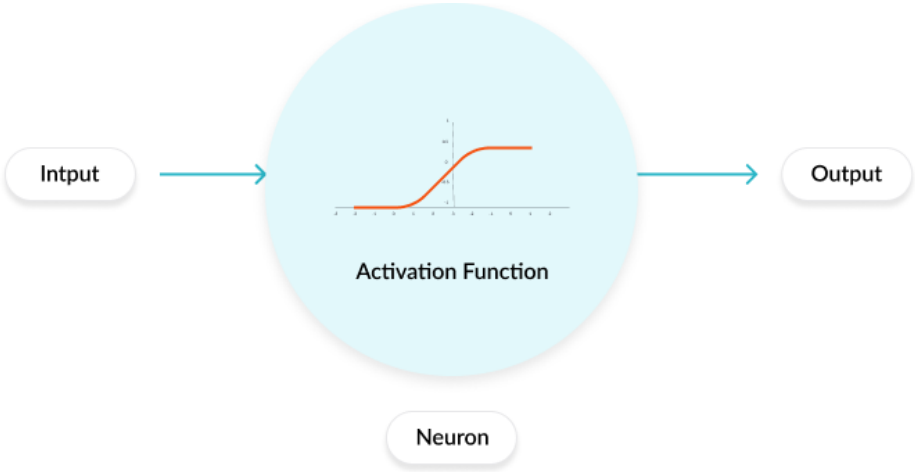

Figure 1.4: Simplified structure of an activation function of an artificial neuron**[21]**.

In a neural network, inputs, which are usually real values, are fed into the neurons in the network. Each neuron has a weight, and the inputs are multiplied by the weight and fed into the activation function. The activation function is the function that makes it possible to process the information that arrives at an artificial neuron in machine learning, as those of the brain do with the electrical signals they receive.

The output of each neuron is the input to neurons in the next layer of the network, and so the inputs flow through multiple activation functions until finally, the output layer generates a prediction. Neural networks rely on nonlinear activation functions, the derivative of the activation function helps the network learn through the process of backpropagation. main existing activation functions:

1. **Sigmoid**: It generates values between zero and one. For very high or low values of input parameters, the network can be very slow to achieve a prediction.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (1.1)$$

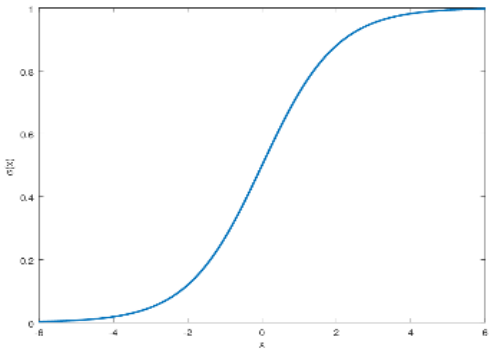

Figure 1.5: Sigmoid function curve.

2. **TanH**: Zero-centered, making it easy to model strong negative, strong positive, or neutral inputs.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (1.2)$$

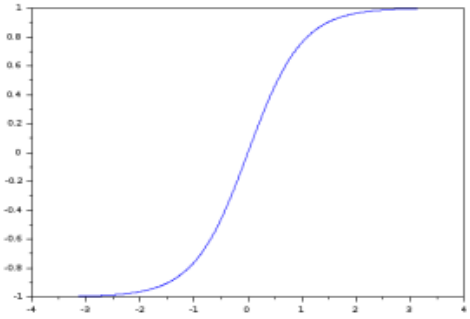

Figure 1.6: TanH function curve.

3. **ReLu**: Very computationally efficient but unable to process inputs that approach zero or negative.

$$f(x) = max(x, 0) \quad (1.3)$$

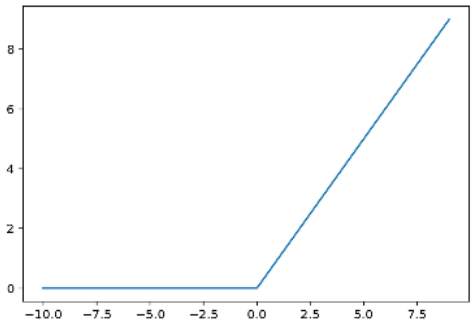

Figure 1.7: ReLu function curve.

3.4 The main architectures of neural networks

A neural network can take different forms depending on the object of the data it processes and according to its complexity and the method of processing the data. The architectures have their strengths and weaknesses which makes them more suitable for a given task, they can be combined to optimize the results. The choice of architecture is thus crucial, determining the objective to be achieved is the best way to choose the most appropriate approach according to its advantages and especially its constraints. In what follows, we present the main architectures of neural networks.

3.4.1 Convolutional Neural Networks (CNN)

Convolutional neural networks, also called convnet (for ”Convolutional Network”), or CNN (for ”Convolutional Neural Network”), have proven to be very effective for tasks involving closely related data, mainly in the field of vision. by computer. There are two parts in a CNN, a first part which is called the convolutional part of the model and the second part, which we will call the classification part of the model which corresponds to an MLP (Multilayer Perceptron) model**[22]**. It is a multilayer neural network and more precisely it is a deep network composed of four types of layers: the convolution layer, the pooling layer, the ReLU correction layer and the fully connected layer.

- **Convolution layer**: this is the most important layer and the heart of the constituent elements of the convolutional network, and it is also the one that performs the most heavy calculations. Its purpose is to identify the presence of a set of features in the images received as input.

- **Pooling layer**: is often placed between two convolution layers: it receives several feature maps as input, and applies the pooling operation to each of them. The pooling operation consists in reducing the size of the images, while preserving their important characteristics.
- **The ReLU correction layer**: therefore replaces all the negative values received as inputs with zeros. It acts as an activation function.
- **The fully-connected layer**: always constitutes the last layer of a neural network, convolutional or not, it is therefore not characteristic of a CNN. This type of layer receives a vector as input and produces a new vector as output. For this, it applies a linear combination then possibly an activation function to the values received as input.

The following figure shows the architecture of a CNN type neural network:

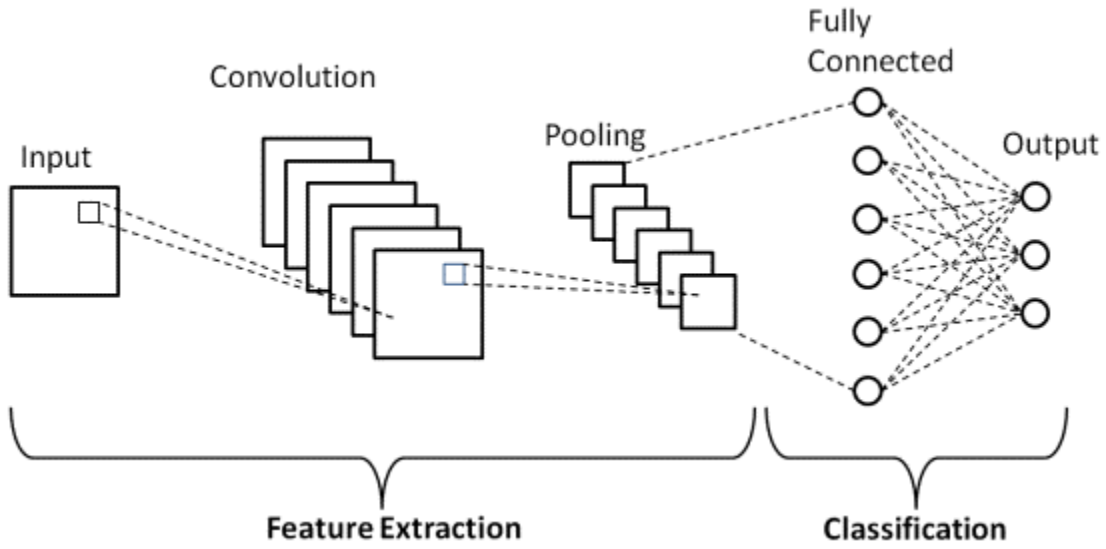

Figure 1.8: How a convolutional neural network works**[23]**.

CNN-type neural networks are used in particular in:

- Facial recognition.
- Identify and classify everyday objects in pictures.

- Powering vision in robots and autonomous vehicles.
- Recognize scenes and suggest relevant captions.

3.4.2 Recurrent Neural Networks (RNN)

A recurrent neural network (RNN) is a type of feedforward neural network that can handle varying sequences. RNN can handle time series because it has a recurrent hidden state whose activation is dependent on the activation of the prior time. Long short-term memory units (LSTMs) are a form of RNN that allows each recurrent unit to adapt to multiple time scale dependencies.

Figure that follows shows an unwrapped RNN into a full network. By unwinding, we simply mean that we are writing the network for the complete sequence. For example, if the sequence we are interested in is a 5-word sentence, the network will be unrolled in a 5-layer neural network, one layer for each word.

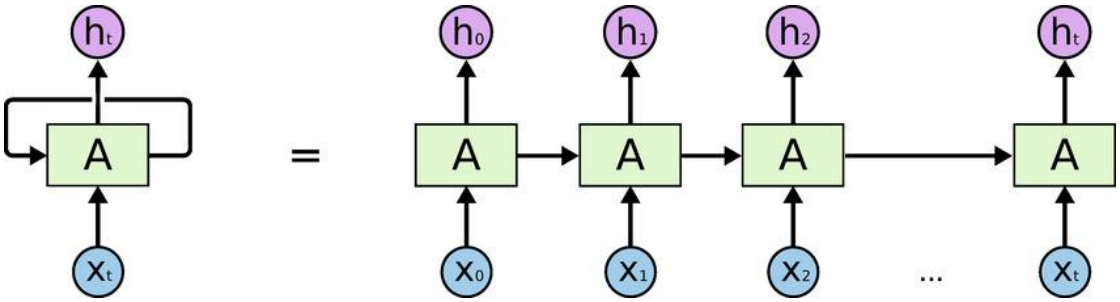

Figure 1.9: Unrolled neural network**[24]**.

An RNN examines a series of inputs over time, X0, X1, X2, up to Xt. For example, it could be a sequence of images in a video or words in a sentence. The neural network has a layer of neurons for each input. When the RNN learns, it backpropagates through time (BPTT), a form of multilayer backpropagation. BPTT uses the chain rule to go back from the last time step (Xt), gradually to each previous step, each time using gradient descent to discover the best weights for each neuron, and also learn the optimal weights that govern the transfer of information between one time step to another. RNNs are used in particular for:

- **Language modeling and text generation**: Models work by predicting the most appropriate next character, next word, or next phrase in a string of text.

- **Automatic translation**: the text in the source language is entered in batches and the model attempts to generate the corresponding text in the target language.
- **Voice recognition**: the input consists of acoustic signals analyzed from an audio recording; the model produces the most probable language phonetic element for each part of the recording.
- **Generation of image captions**: the input is an image and the model identifies elements of the image and generates text that describes it.
- **Time-series anomaly detection**: The input is a series of sequential data, such as a series of events in a potential cybersecurity incident. The model predicts whether the data is an anomaly with respect to the normal behavior.
- **Video markup**: The input is a series of video frames and the model generates a textual description of each frame in the video.

3.4.3 Autoencoder models (AE)

A neural network called an autoencoder can be used to learn a compressed representation of raw input. An autoencoder is made up of two sub-models: an encoder and a decoder. The encoder compresses the input, while the decoder attempts to reconstruct it from the encoder's compressed form**[25]**. The encoder model is saved after training, whereas the decoder is deleted.The encoder can then be used as a data preparation approach to extract features from raw data so that a new machine learning model can be trained. The following figure schematizes a simple autoencoder, whose encoder processes images (inputs), in order to represent them as points in a two-dimensional space (encoded representation), then decodes this representation, in order to find the starting data (output).

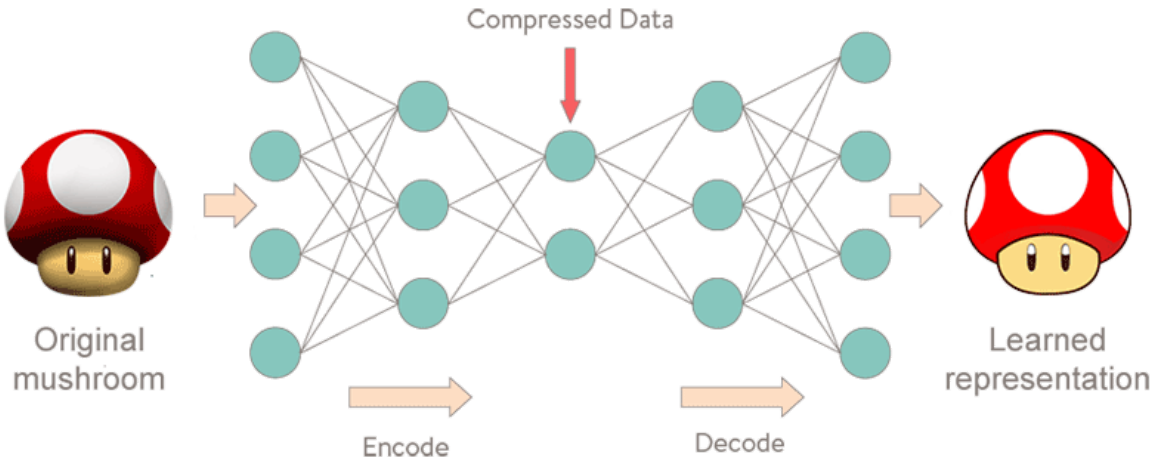

Figure 1.10: Autoencoder**[26]**.

3.5 Reinforcement learning (RL)

Reinforcement Learning is a type of Machine Learning approach in which the agent is given a delayed reward in the following time step in order to evaluate its prior activity. It was largely employed in video games (e.g., Atari, Mario), with human-like or even superior performance. As the method advances with the use of Neural Networks, it is now capable of tackling more complicated problems**[27]**.

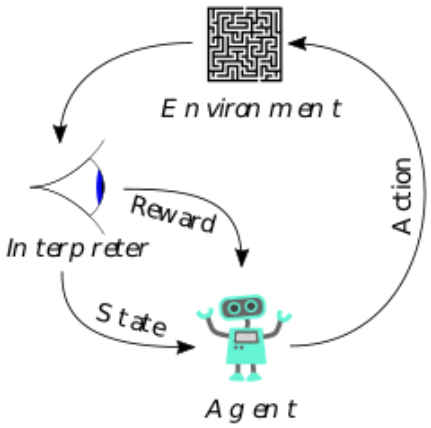

Figure 1.11: principle of reinforcement learning.

3.6 Advantages and disadvantages

Neural networks have been widely used in recent years in many fields because they have more advantages than disadvantages. We list its advantages and disadvantages below.

Advantages

- Often high precision.
- Robust, works when the learning set contains noise.
- Allows complex problems to be solved by machine learning (no need to manually define AI rules as in the classic method).
- The output can be a discrete, continuous value, or a vector of several discrete or continuous values.
- Adapts well to problems that require high levels of abstraction that are difficult to describe explicitly with the classical method (code all treatments and conditions).

Disadvantages

- Long learning curve (due to high number of parameters).
- Difficult to understand the model.
- Non-evidence of the choice of hyperparameters.

Conclusion

Machine learning and deep learning are the two most important concepts that make artificial intelligence possible. Machine learning that learns based on a large set of structured data. Deep learning does not need structured data, but systems that work from several layers of neural networks without being guided by humans. We have seen in this chapter what deep learning is and its fields of application, with the important notions that are related to (definition, architectures, etc.), artificial neural networks and deep neural networks . With some platforms among the fields of application of deep learning are recommender systems, we describe this area in the third chapter, because it is the context of our work.

Chapter 02

Recommender systems

Introduction

Nowadays, a large number of recommendation systems (RS) are used in various fields. The main objective of these systems is to filter the flow of information to provide each user with resources that meet his information needs. In order to meet these constraints, the engines of these systems manage users profiles to choose which resources to transfer to each of them, and update these files according to user feedback and interactions. The recommendation system provides users with suggestions that correspond to their information needs and preferences. These last can be found across a variety of industries, companies, financial services, online music/radio, television, video, online publications and many more.

1. Recommender systems history

Recommender systems roots can be traced back to extensive work in cognitive science, approximation theory, information retrieval, foresight theory also has links to management science and marketing, in modeling consumer choice**[28]**.

The recommendation systems are recognized quite early in the history of computer science, Information Lens System**[29]** can be considered as the first recommendation system at the time, the most common approach to the problem of sharing information in the email environment was the interest-based distribution list.

During the 90s, recommender systems became an important area of research with the publication of the first articles in the field of collaborative filtering. The academic literature has introduced the term collaborative filtering by the Tapestry system**[29].** It was developed in 1992 by the research center of ”Xerox” in the United States and which allowed users to create permanent queries, based on users tags. The definition for filtering was also given by Malone :”Even though the term has a literal

connotation of leaving things out (negative filtering: removal), we use it here in a more general sense of selecting things from a larger set of possibilities (positive filtering: selection)” **[30].** A

few years later, a number of academic recommendation systems emerged in 1994 and 1995, such as the news and movie recommendation system developed by ”GroupLens”**[31]** and the ”Ringo” music recommendation. These two systems are also based on collaborative filtering, books , videos, movies, web pages, Usenet news articles and Internet links. Subsequently, with the rise of the Internet and Web applications, there has been an enthusiasm for recommender systems that have developed in different fields, We can cite:

- Movie recommendation systems, such as: Movielens and Eachmovie.
- Book recommendation systems (Bookcrossing).
- Music recommendation systems (LastFM).
- News article recommendation systems.
- Joke recommendation systems (Jester).
- Recommendation systems introduced on e-commerce sites (Amazon).
- Restaurant recommendation systems.
- Recommendation systems integrated into documentary Extranets (the Extranet Credit Agricole documentary).
- Recommendation systems integrated into search engines (the search engine AOL search).
- Recommendation systems implemented on recruitment sites (JobFinder).
- Bibliographic citation recommendation systems.

2. Definition

Recommender systems can be defined in many ways, given the diversity of classifications proposed for these systems, but there is a general definition by Robin Burke**[32]** which defines them as follows: ”Systems capable of providing customized recommendations to guide the user to interesting and useful resources within a large data space”.

It is also defined as: a filter of incoming information flows customized for each actor**[33]**. In other words, in order to personalize the search for information in a particular field of application, a filtering system collects, selects, classifies and suggests to the user information which probably meets his long-term interests.

Two basic entities that appear in all recommender systems are the user and the item. The ”user” is the person who uses a recommendation system, gives their opinion on various articles and receives new recommendations from the system. “Item” is the general term used to refer to what the system recommends to users. The input data for a recommender system depends on the type of filtering algorithm used.
Generally, they belong to one of the following categories:

- The ratings: (also called the votes), express the opinion of the users on the articles(example: 1 bad to 5 excellent).
- Demographic data: refers to information such as age, gender, country and level of education of users. This type of data is usually hard to obtain and normally collected explicitly.
- Content data: which is based on a textual analysis of the documents related to the items evaluated by the user. The characteristics extracted from this analysis are used as inputs in the filtering algorithm in order to deduce a user profile.

To perform the filtering, the recommendation system uses the profiles representing relatively stable user preferences to calculate recommendations. This calculation is done by predicting the scores that a user is likely to attribute to the items. The RS adapts this profile over time by making the most of the relevance feedback that users provide on the items received. For example, in Figure below, the decision function of the system processes the incoming flow of items to suggest to the user, by consulting his profile, the items he prefers. In turn, the user can provide ratings, i.e. frequently rate the recommendations, so that the system better understands their information needs, and consequently provides them with better new recommendations.

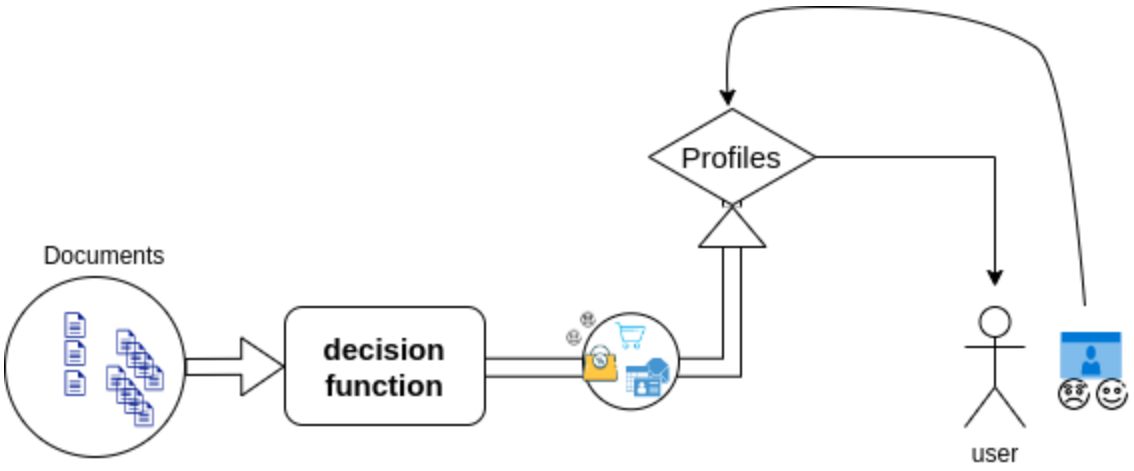

Figure 2.1: General diagram of an information filtering system (document)**[34]**

4. Objectives of recommender systems

We previously defined recommender systems as tools and methods that provide users with suggestions of items they would like to buy or use.

In this section, we aim to give a more fine-grained definition by illustrating the possible uses of an RS. The first distinction that needs to be made is between the RS user and the service provider**[35]**. For example, a restaurant or hotel recommendation system is usually used by an intermediary (e.g. TripAdvisor) in order to increase its conversion rate, i.e. increase the number of people going to a given restaurant or book more hotel rooms. On the other hand, the user's motives for using a system like TripAdvisor are to find a restaurant or hotel that matches their tastes and needs, thus increasing their satisfaction. In fact, there are several reasons why service providers make use of recommendation engines:

- **Increase revenue**: In other words, it means increasing the number of items that are sold. This feature is probably the most important in an RS context.

The goal here is to actually sell more items than there would have been without any recommendation. To achieve this goal, the system recommends items that are expected to satisfy the user's tastes and needs. However, we need to make a distinction between predicting users' interests in an item and the likelihood that users will actually choose/select the recommended item.

- **Increase the diversity of sold items** : The purpose of this feature is to encourage users to select items that would remain unknown without a recommendation.

For example, in the case of a book RS (e.g. Amazon bookstore), the service provider wants to be able to sell books from all of its catalogs and not just the 5 most popular.

- **Understanding users**: Another important feature of an RS is being able to describe user preferences. These preferences can be gathered explicitly or by predicting them. This data can be used by the service provider to better manage its production or stock.

Now we explain the functions users might be interested in when using an RS.

- **Ranking a list of items**: This is probably one of the most important functions for an RS, i.e. to deliver certain good items to the current user, according to rating predictions. In other words, recommending items that the user should like.
- **Recommend Sequence**: This feature is more about adapting to users' long-term preferences. The principle is to generate a coherent series of recommendations instead of providing a succession of independent ones. For example, it would make sense to recommend Matrix 2 Reloaded after recommending Matrix 1.
- **Improved browsing**: Given a large catalog, the task of an RS can improve the user's browsing experience by helping them find items that match their tastes and needs.

5. Phases of recommendation process

A recommendation system generally requires three (03) steps, which are illustrated in the following figure:

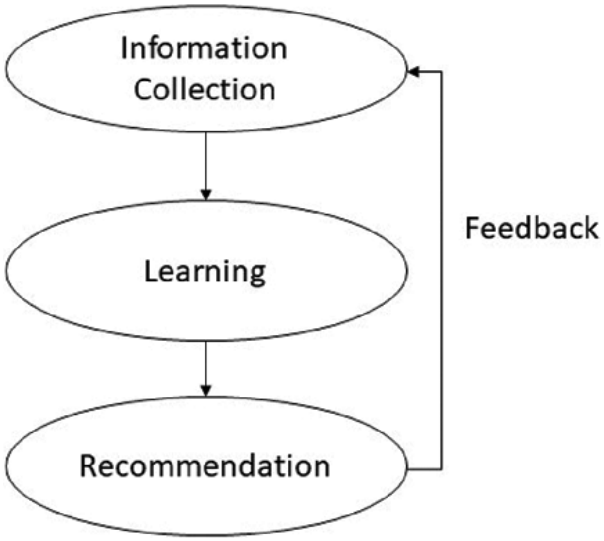

Figure 2.2: Main phases of recommendation process**[36]**.

5.1 Information collection

To be relevant, a recommender system must be able to make predictions about users' interests. It is therefore necessary to be able to collect a certain amount of data on them in order to be able to build a profile for each user. A distinction can be made between 2 forms of data collection:

1. **Explicit data collection** - (Active filtering): The collection relies on the user explicitly telling the system about their interests.

Example: Ask a user to comment, tag/label, rate, like or even add as favorites content (objects, articles, etc.) that interests him. We often use a scale of ratings ranging from 1 star (I don't like it at all) to 5 stars (I really like it) which are then transformed into numerical values in order to be used by the recommendation algorithms.

Advantages: ability to reconstruct the history of an individual and avoid aggregating information that does not correspond to this unique user (several people on the same workstation). **Disadvantage**: The information collected may contain reporting bias**[37].**

2. **Implicit data collection -** (Passive filtering): It is based on an observation and analysis of user behavior carried out implicitly in the application that embeds the recommendation system, all done in ”background” (basically without asking the user). Exemple :

- Obtain the list of items that the user has listened to, watched or purchased online.
- Analyze the frequency of consultation of content by a user, the time spent on a page.
- Monitor the online behavior of the user.
- Analyze his social network.

Advantage: No information is requested from users, and all information is collected automatically. The data recovered are substantially correct and contain no reporting bias. **Disadvantage**: The data retrieved is more difficult to attribute to a user and may therefore contain attribution bias (common use of the same account by several users). A user may not like certain books they have purchased, or they may have purchased it for someone else.

5.2 Learning

That means applying a learning algorithm to filter and exploit the user’s features from the feedback gathered in information collection phase, in other words the implementation of a matrix called ”user matrix” or ”user model” including the information concerning the users collected during the previous step which is the collection of information. It can be represented as a table that contains data collected about the user associated with the products available on the website. User interests generally change over time. which makes it another important point is how time influences the user profile. The data of the user model should therefore be constantly readjusted to remain in line with the new centers of interest of the user. The figure below represents a user matrix where each cell with a number in it is the rating given by some user on a specific item, while those marked with question marks are unknown ratings that need to be predicted. In some other literature, this problem may be named collaborative filtering, matrix completion, matrix recovery, etc.

	Item 1	Item 2	Item 3	...	Item n
User 1	2	3	?	...	5
User 2	?	4	3	...	?
User 3	3	2	?	...	3
...	...	...	...	...	...
User m	1	?	5	...	4

Figure 2.3: Example of rating matrix**[38]**.

5.3 Recommendation

To extract a list of suggestions from a user model, the algorithms use the notion of measuring similarity between objects or people described by the user model. Similarity aims to give a value or a number (in the mathematical sense of the term) to the resemblance between 2 things. The stronger the resemblance, the greater the value of the similarity. Conversely, the weaker the resemblance, the lower the value of the similarity**[39]**.

6. Recommender system classifications

Many approaches and methods for classifying recommender systems have been proposed. Sometimes various terminologies are used to designate the same method or approach. This is why some researchers have taken an interest in the classification of these methods and are proposing a unified terminology as shown in Figure down Here are some classifications:

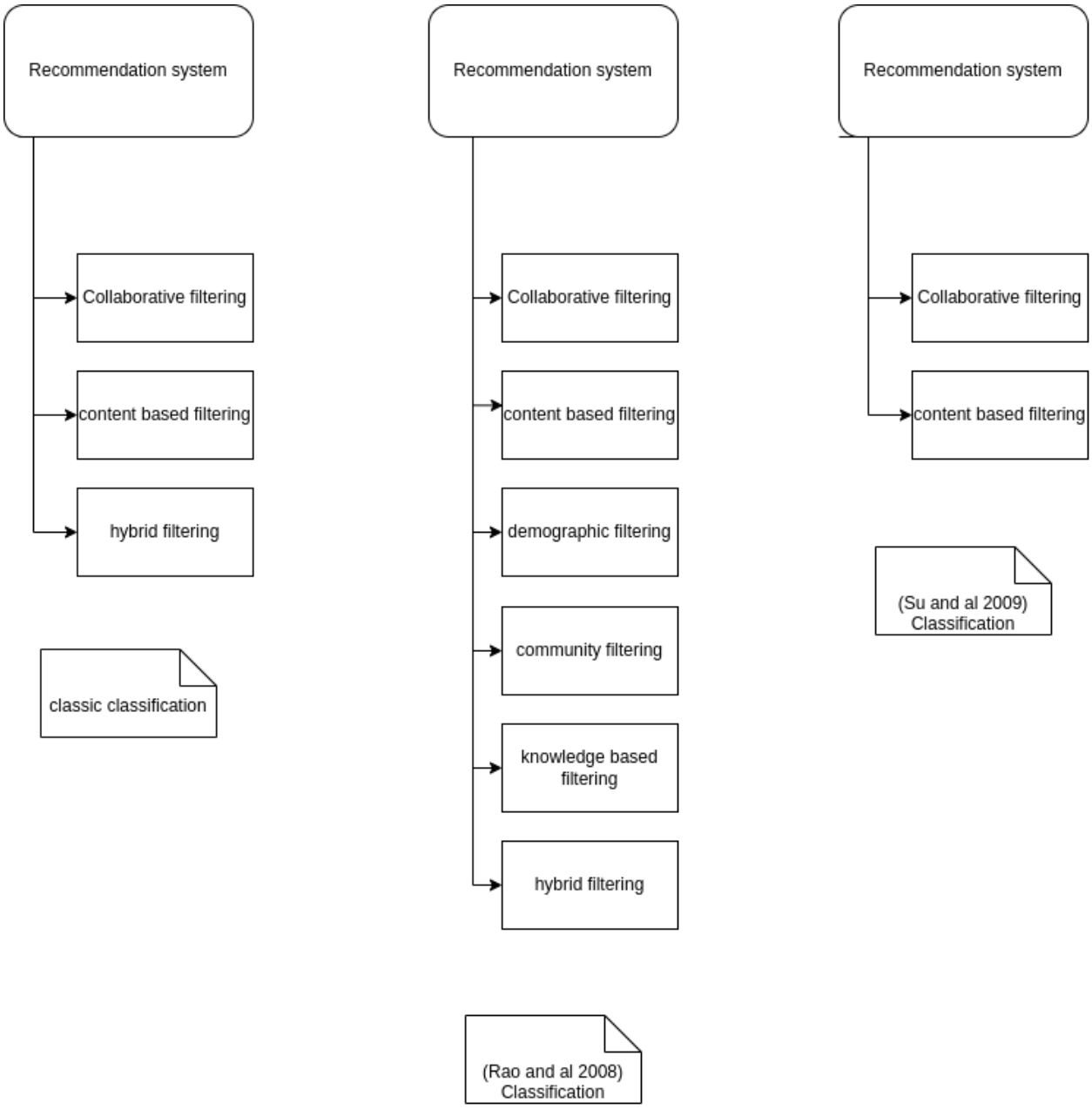

Figure 2.4: Major Classifications of Recommender Systems.

6.1 Classic classification

This classification is recognized by three types of filtering.

- Collaborative filtering (CF).
- Content-Based Filtering (CBF).

- Hybrid filtering.

6.2 Su Classification

It is used in collaboration systems. They propose a sub-classification that includes hybrid techniques classifying them in hybrid collaboration methods.**[40]** classifies collaborative filtering into three categories:

- Memory-based CF approaches: for K-nearest neighbors.
- Model-based CF approaches encompass a variety of techniques such as: clustering, Bayesian networks, matrix factorization, Markov decision processes.

6.3 Burke's Classification

Burke**[41]** offers a very complete classification of existing recommendation techniques by identifying the input data of each method and its algorithm used. It defines five types of Recommendation Techniques:

- Collaborative filtering.
- Content-based filtering.
- Demographic filtering.
- Knowledge based filtering.
- Community screening.

7. Recommendation techniques

Currently, there are three major filtering approaches: content-based, collaborative, and hybrid [Cacheda et al, 2011]**[42]**. Content-based filtering compares new documents to each user's profile, and recommends those that are closest. Collaborative filtering compares users with each other based on their past judgments to create communities, and each user receives the documents deemed relevant by their community. Hybrid filtering combines content-based filtering and collaborative filtering to maximize the benefits of each.

7.1 Collaborative filtering

Collaborative filtering methods are based on collecting and analyzing a large amount of information about users' behaviors, activities and preferences and predicting what users will like based on their similarity to other users. The main advantage of the collaborative filtering approach is that it does not rely on the parsable content machine and is therefore able to accurately recommend complex items such as movies, without requiring an understanding of the item itself.

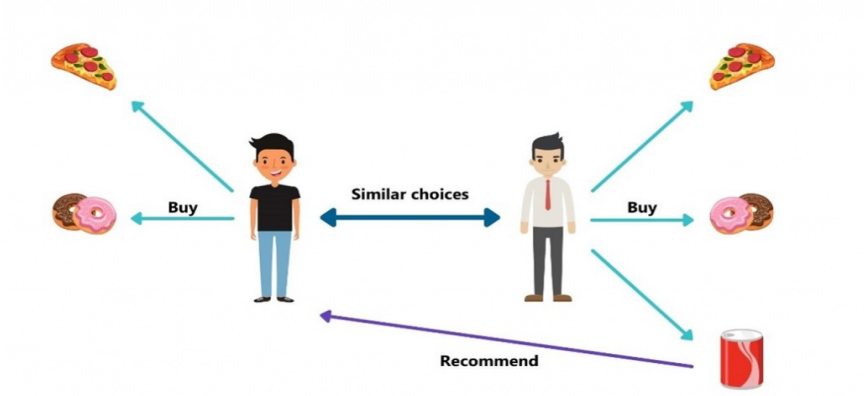

Figure 2.5: The principle of collaborative filtering**[43]**.

Two forms of collaborative filtering are :

1. **User-based** : User-based collaborative filtering (UBCF) relies on the assumption that users who have similar ratings will have similar preferences. The first step in UBCF is to create "neighborhoods" of similar users. The mathematics behind creating a similarity matrix can get complicated but often rely on correlation. Then, the average rating of the neighborhood for a particular item becomes the predicted rating for a user in that neighborhood without a rating for that item. The item with the highest predicted rating would then be recommended to that user.

2. **Item-based** : Item-based collaborative filtering (IBCF) is very similar. Recommendations here are based on the similarity between items instead of users. So, the assumption is that a user will prefer items that are similar to the items that they have already rated highly. Just as before, a similarity matrix is created between all the items. The predicted rating for an item for a particular user is the weighted sum of the known ratings from that user for similar items.

Advantages

- Does not require any knowledge of the content of the item or its semantics.
- The quality of the recommendation can be evaluated.
- The higher the number of users, the better the recommendation.

Disadvantages

- Cold start.
- New Item.
- New user.Confidentiality issue.
- Complexity: in systems with a large number of items and users, the calculation increases linearly.

7.2 Content-based filtering

In this approach, also called cognitive filtering, the choice of documents to recommend is based on a comparison of the topics covered in the documents with the topics of interest to the user. Content-based filtering methods are based on an item description and user preference profile. In a content-based recommendation system, the keywords that are used to describe items. Next, the user profile is designed to indicate the type of item that the user likes. In other words, these algorithms were trying to recommend items that are similar to ones the user has liked in the past (or is reviewing in the present). In particular, various candidate items are compared with items previously rated by the user and the most matching items are recommended. This approach has its roots in information retrieval and information search filtering.

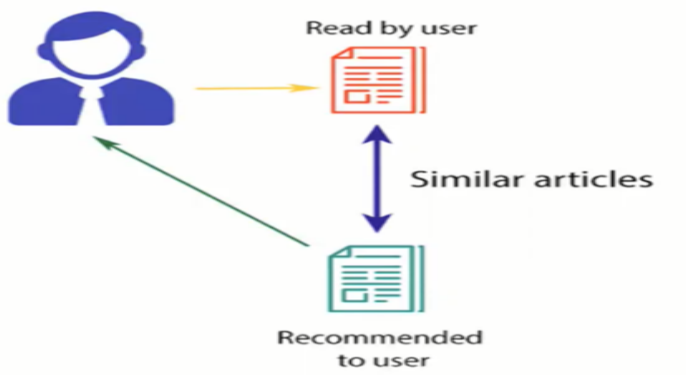

Figure 2.6: The principle of content based filtering**[44]**.

Advantages:

- No need for a large community of users to be able to make recommendations.
- A list of orders can be generated even if there is only one user.
- Quality grows with time. No need for information about other users.
- Take into consideration the unique tastes of users.

Disadvantages:

- Content analysis is necessary to make a recommendation.
- Problem of recommending images and videos in the absence of Metadata.
- Need for user profile**.**

7.3 Hybrid Filtering

Many systems are based on the combination of these two previous approaches. By integrating collaborative filtering features, content-based filtering produces a synergistic effect. How these two approaches fit together varies, but both have complementary strengths.

Documents can then be routed to other users using both collaborative (rating) and content based (document content) filtering criteria**[42]**.

More generally, hybrid systems manage content-oriented user profiles, and the comparison between these profiles gives rise to the formation of user communities enabling collaborative filtering.

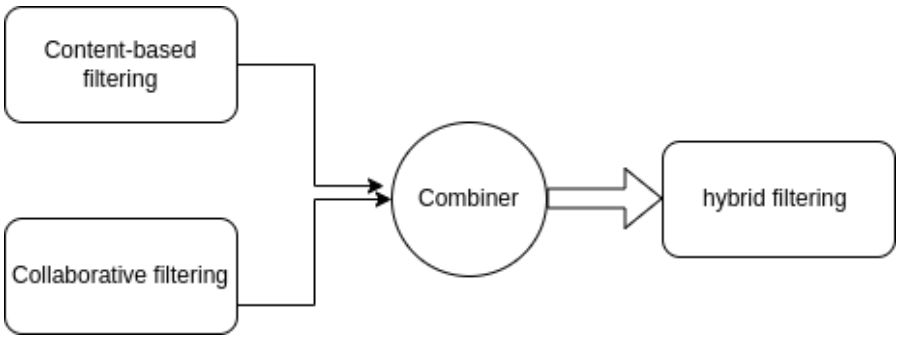

Figure 2.7: The principle of hybrid filtering.

8. Pros and cons of recommender systems

- Sometimes, RS suggests wrong products and services by analyzing little or incomplete user information.
- Customers can make wrong choices, if biased recommendations are made by RS.
- an item must have enough ratings for it to be considered in the recommendation Process.
- Cold start,a new user who has not yet accumulated enough ratings cannot have relevant recommendations.
- As the database of ratings grows, the recommendation becomes more accurate, and a small database leads to less accurate recommendation.
- It helps in increasing the average order value by displaying personalized options.
- It helps in transforming service providers or sellers to clients.
- It helps in increasing the number of items per order, because clients get more choices when browsing.
- It helps in engaging shoppers, and brings traffic to your site.

Conclusion

This chapter aimed to provide a non-exhaustive overview of recommender systems. We have discussed their history, a theoretical definition of what a recommender system is. also the objectives of RSs and main phases of the recommendation process, after that we have presented the three main types of recommender systems: collaborative filtering ,content-based filtering, and finally hybrid approaches. Next, we presented the Deep learning techniques in recommender systems.

Chapter 03

Machine learning and deep learning techniques in recommender systems

In this section, we present some state-of-the-art work based on deep learning and machine learning for recommender systems

3.1 Collaborative filtering using nearest k-neighbors (KNN)

Collaborative neighbor-based filtering algorithms typically use the entire matrix of user notes to make the recommendation. To use the KNN algorithm, the system must have a measure of similarity to distinguish between users who are close and those who are far away. This measure of similarity can be derived from the Euclidean distance, the cosine measure, the Pearson correlation, etc. Then, when the system is queried on new information, it will find the k users that are closest to the target user. Finally, a majority vote is taken to decide which item will be recommended (user-based filtering).

3.1.1 Calculation of similarity

The purpose of calculating similarity is to determine the extent to which two users are similar. There are several ways to calculate this similarity, however the most commonly used methods are:

- **The measurement of cosines**

In this method users A and B are considered as two vectors of the same origin in a space of n dimensions, n is equal to the number of items evaluated by both users. The more similar two users are, the greater the angle between their vectors respective is small. Empirically, the similarity between these two users is calculated by the following Cosine formula:

$$Sim(A,\ B) = \frac{\sum_{i=1}^{n} \nu_{Ai} \times \nu_{Bi}}{\sqrt{\sum_{i=1}^{n} \nu_{Ai}^{\ 2}} \times \sqrt{\sum_{i=1}^{n} \nu_{Bi}^{\ 2}}} \qquad (3.1)$$

n: Number of common items between A and B voted by ***v***

vAi: Vote of A for item *i*

vBi: B's vote for item *i*

- **Pearson correlation**

The Pearson correlation is a method derived from statistics. It is also widely used in the field of recommender systems to measure the similarity between two users. The following formula presents the calculation of the similarity between A and B with this measure.

$$Sim(A,\ B)\ =\ \frac{\sum_{j}(\nu_{Ai}-\overline{\nu_{Ai}})(\nu_{Bi}-\overline{\nu_{Bi}})}{\sqrt{\sum_{j}((\nu_{Ai}-\overline{\nu_{Ai}})(\nu_{Bi}-\overline{\nu_{Bi}}))^{2}}} \quad (3.2)$$

i: Number of objects voted by both A and B.

ν_{Ai}: Vote of A for item *i*.

$\overline{\nu_{Ai}}$: Average vote of A.

ν_{Bi}: B's vote for item *i*.

$\overline{\nu_{Bi}}$: Average B votes.

Once all the similarities of the target user A with respect to the other users are calculated and the n most similar users who constitute the neighborhood of this target user are defined, the prediction of the value of an item j evaluated by user A (PA,j) is calculated using the weighted sum of the estimates of the nearest neighbors who have already estimated item j:

$$P(A,j)\ =\ \frac{\sum_{i=1}^{n} Sim(A,i)\times \nu_{i,j}}{\sum_{i=1}^{n} Sim(A,i)} \quad (3.3)$$

n: Number of users present in the vicinity of A, having already voted on item j.

$\nu_{i,j}$: Vote of user i for object j.

Once the similarity has been calculated, the prediction (see Formula 3.3) is calculated to predict the target user's score for an item.

3.2 Multilayer Perceptron

A multi-layered perceptron (multilayer perceptron or MLP) is a perceptron that combines with additional perceptrons, stacked in multiple layers, to solve complex problems. In the case of recommendation systems, multilayer perceptrons are used to encode users and items separately before combining them with another multilayer perceptron. They model interactions between users and items, and select the most representative items for each user and the most important item attributes for each user.

To model the user-item interaction, we use a multilayer representation, where the output of a layer serves as input to the next layer. The first input layer has two Vu and Vi input vectors that represent the user u and item i. After the input layer, there is an integration layer. This layer is fully connected, which projects the separate representation into a dense vector. These integration layers are then introduced into a multi-layered neural architecture to map latent vectors to prediction scores. We can also customize each hidden layer to discover new latent structures from user-item interactions. The last layer gives the expected score **[45]**.

3.3 Recommender system using Bayesian Personalized Ranking (BPR)

When users buy online, they usually only browse the first pages of websites. In addition, more and more people are using tablets and mobiles to shop online, so it is essential to make a personalized ranking.

When a user clicks on an item to see its details, it usually means that some features of the item are attractive. Thus, it can be assumed that users prefer these items to others and use them to classify them after a user has browsed the website for a period of time. The main task of the custom ranking is to provide a user with a classified list of items **[46]**.

Let "U" all users and "I" all items. Figure 3.1 below shows how implicit data are processed for item recommendations.

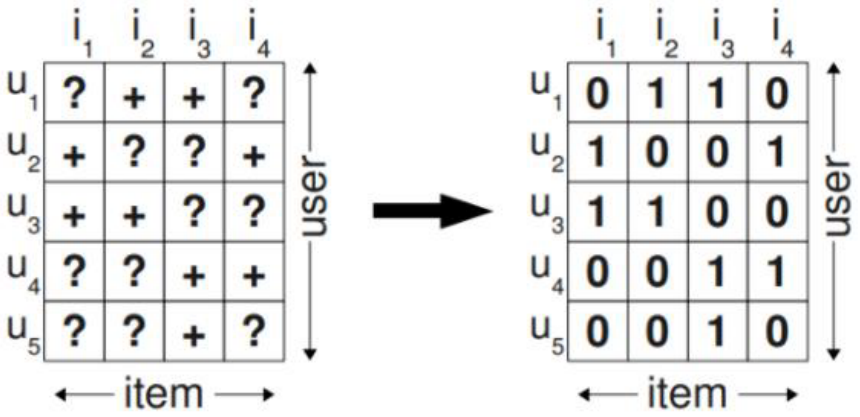

Figure 3.1: The existing interactions between the user and the item **[46].**

The usual approach is to predict a custom Xui score for an item that reflects the user's preference for the item. After that, the items will be ranked according to this score. Here, as you can see in the figure above, all existing interactions between the user and the item are labeled as a positive class (1) and the rest of the interactions are labeled as a negative class (0).

In the BPR **[46]** approach, instead of taking an item, pairs of items are considered training data. Optimization would be done based on the ranking of these user-item pairs instead of just noting the user-item interaction. The data set that would be considered is formulated as follows:

$$(u, i, j) \in D_s$$

The semantics of (u, i, j) $\in D_s$ is that the user u is supposed to prefer i to j.

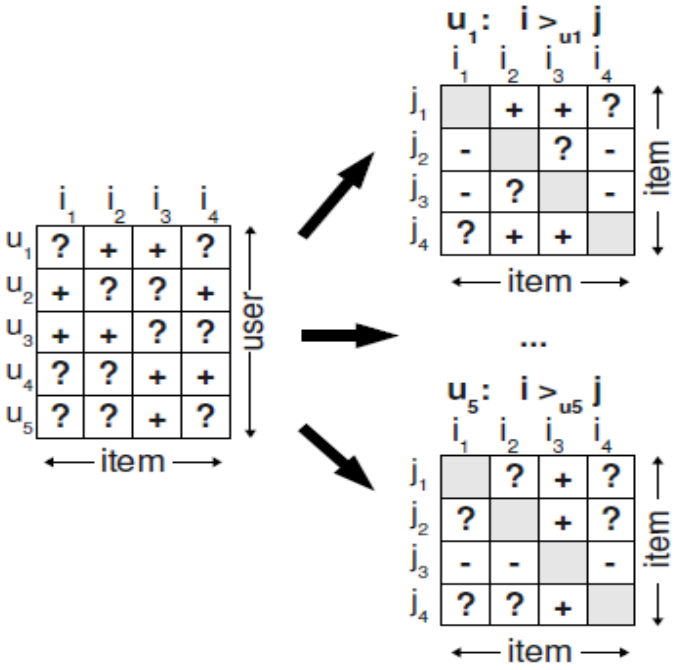

Figure 3.2: Preferences per pair specific to the user between a pair of items **[46].**

In figure 3.2 above, user (u1) saw item (i2) but not item (i1), so the algorithm assumes that this user prefers item (i2) to (i1) (i2 > i1) and gives a positive sign. No inference can be made on a preference for items that have both been seen by a user and are displayed as "?". The same applies to two items that a user has not yet seen (for example, item (i1) and (i4) for user (u1)). On the contrary, you can see a negative sign for (i1, j2) because the user prefers item 2 (i2) to item 1 (i1).

The authors here propose that BPR consists of the BPR-OPT optimization criterion and the Learn BPR algorithm for optimization. Rather than predicting a specific score for each item, we only need to predict the relative preferences of the user for all pairs (i, j).

BPR-OPT :

$$\sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda\|\Theta\|^2 \qquad (3.4)$$

Or σ is the sigmoid (see Formula 1.1) logistics function:

$\hat{x}_{uij}(\Theta)$ in the above equation is a real value function that represents the relationship between user u, item i and item j and is generally calculated using the matrix factoring model.

Θ: represents the parameter vector of an arbitrary model class (for example, matrix factorization).

$\lambda\ \Theta$:are model-specific adjustment parameters We can formulate the individual probability that a user prefers item i to item j as follows:

$$p(i>uj|\Theta):=\sigma(\hat{x}_{uij}(\Theta))$$

Here, > u is the preferred but latent structure for the user u. It is also important to note that p (> u | θ) is a user-specific likelihood function.

Likelihood function: is a function of the parameters of a statistical model calculated from observed data.

$$\mathbf{P\ (\Theta|>u)}\ \alpha\ \mathbf{p\ (>u\ |\Theta)\ p\ (\Theta)} \qquad (3.5)$$

We must therefore maximize the number of correct predictions of all pairs (i,j), which is equivalent to maximizing the AUC which is the area under the ROC curve.

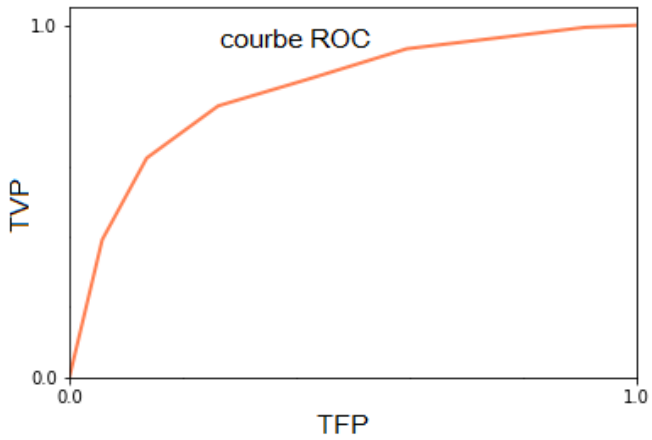

Figure 3.3: the ROC curve **[47].**

A receiver operating characteristic (ROC) curve is a graph representing the performance of a classification model for all classification thresholds. This curve plots the rate of true positives as a function of the rate of false positives:

The true positive (TVP) rate is the equivalent of a recall. It is therefore defined as follows:

$$TVP = \frac{VP}{VP+FN} \tag{3.6}$$

The false positive (**TFP**) rate is defined as:

$$TFP = \frac{FP}{FP+VN} \tag{3.7}$$

Area Under The Curve:

The above equation is the AUC formula, where I is the set of all items, and Iu+ is the set of items that the user clicked on:

$$\text{AUC} = \frac{1}{|U|}\sum_{u\in U}\frac{1}{|I_u^+||I/I_u^+|}\sum_{i\in I_u^+}\sum_{j\in|I/I_u^+|}\delta(\widehat{x_{uij}} > 0) \tag{3.8}$$

The AUC is the area under the ROC curve that is plotted with a true positive rate as a y axis and a false positive rate as an x axis for different thresholds of the model that is considered. In other words, for a set of item recommendations, we can plot the ROC curve by calculating the percentage of good item recommendations and the percentage of bad recommendations for each threshold.

3.4 Neural Collaborative Filtering (NCF)

Matrix factorization matrix (MF) for recommendation systems is one of the methods used by online services such as Netflix to accelerate the search for content recommendations for users

NCF abbreviation of Neural Collaborative Filtering, has attempted to achieve the following objectives:

- NCF tries to express and generalize matrix factorisation (MF) within its framework.
- NCF tries to learn user-item interactions via a multi-layer perceptron.

Collaborative neural filtering uses MF to identify the relationship between items and users. User assessments on items at entry help predict how users would evaluate items so that users can get the recommendation based on prediction**[48]**.

Collaborative Neural Filtering (NCF) aims to design a better dedicated interaction function to model the interaction of latent features between users and items by:

1. Modeling the interaction of user-item features via the neural network architecture using a multi-layer Perceptron (MLP). This is an upgrade to MF because MLP can (theoretically) learn any continuous function and has a high level of non-linearities (due to multiple layers).
2. Generalizing and expressing MF as a particular case of CCN. MF is very successful in the area of recommendations, this will give more credibility to NCF.

3.5 Collaborative Auto-encoder Based Filtering (ACF)

Auto-encoders are an unsupervised learning technique in which we exploit neural networks for the task of learning representation by attempting to reconstruct its input data in the output layer. They consist of an input layer, a hidden layer and an output layer. The input data is transmitted to the input layer. The entrance layer and the hidden layer build an encoder. The hidden layer and the output layer build a decoder. The output data comes out of the output layer **[49].**

There are two general ways to apply auto-encoder to the referral system:

- Use auto-encoder to learn representations of entities smaller than the bottleneck layer
- Fill the interaction matrix blanks directly into the reconstruction layer. Almost all autoencoder variants can be applied to the recommendation task.

There are many automatic encoder variants currently used in referral systems. The four most common are:

A)The Denoising Autoencoder (DAE)

Corrupts the entries before mapping them into the hidden representation, then rebuilds the original entry from its corrupted version. The idea is to force the hidden layer to acquire more robust features and prevent the network from simply learning the identity function.

B)Stacked Denoising Autoencoder (SDAE)

Stacks several auto-encoders to get higher-level representations of the inputs. The training is generally optimized with greedy algorithms, going layer by layer. The apparent drawbacks here are the high cost of calculating the training and the lack of scalability for large-scale features.

C)Marginalized Denoising Autoencoder (MDAE)

Avoids the high calculation cost of SDAE by marginalizing the corruption of stochastic features. Thus, it has a fast drive speed, simple implementation and scalability to large data.

D) Variational Autoencoder (VAE)

Is an unsupervised latent variable model that learns in-depth representation from large-scale data. The idea is to encode the input as a probability distribution rather than a point estimate as in the vanilla auto-encoder. Next, VAE uses a decoder to reconstruct the original input using samples from this probability distribution.

AutoRec :

AutoRec One of the first models to consider the problem of collaborative filtering from the point of view of an auto-encoder.

AutoRec takes user r (u) vectors or item r (i) vectors as input and aims to rebuild them in the output layer, it has two variants: AutoRec based on items (I-AutoRec) and AutoRec based on user (U-AutoRec).

NCF is an extension of AutoRec and has the following two advantages:

- Deploys de-escalation techniques, making NCF more robust;

- It incorporates secondary information such as user profiles and item descriptions to mitigate scarcity and cold start influences.

Conclusion

Recommendation systems are a powerful new technique and are rapidly becoming a crucial tool especially for e-commerce on the Web. In this chapter, we have presented the recommender systems that have become ubiquitous in recent years in many areas from the perspective of deep learning and machine learning.

We first defined the concept of Collaborative filtering using nearest k-neighbors (KNN). Then we presented the Bayesian Personalized Ranking (BPR) in the recommender systems. And the Neural Collaborative Filtering (NCF), More specifically, We can say that we've discussed a set state-of-the-art work that exploits machine techniques and deep learning in recommendation systems.

In the next chapter we describe the implementation of a neural collaborative filtering in order to create a book recommendation system. This algorithm is based on deep learning.

Chapter 04

Implementation and experimental results

Introduction

In this chapter, we will talk about the proposed model, the implementation of a books recommender system on BookCrossing dataset, then we will explain the steps followed to preprocess this dataset, then we present the way each part of our project work, after that we will show the model built to train the dataset, in the end we will discuss the results found.

1. Development environment ,programming language, Libraries and frameworks

In this project we have used many tools, frameworks, libraries and new software. We will define all of them briefly.

Python

Figure 4.1: Python Logo

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to

learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed**[50]**

NumPy

Figure 4.2: NumPy logo

NumPy is a scientific package designed for scientific computing with Python language, and it allows using code of some other languages like C/C++ and Fortran, it has N-dimensional array objects, broadcasting functions, and other complex mathematical functions **[51].**

Pandas

Figure 4.3: Pandas logo.

Pandas is an open source Python library designed for data analysis and data structure ease of use. It helps Python users to work on data analysis workflow without thinking to change over other data analysis languages **[52].**

Scikit learn

Figure 4.4: Scikit learn logo.

Scikit-learn (Sklearn) is one of the most used libraries in machine learning, developed by David Cournapeau as a Google summer of code project in 2007**[53]**, Simple and efficient tools for predictive data analysis, accessible to everybody, and reusable in various contexts, Built on NumPy, SciPy, and matplotlib, open source, commercially usable.

TensorFlow

Figure 4.5: TensorFlow logo

Released on the 15th of November 2015 by Google **[54]**, TensorFlow **[55]** is the newest open source library written in Python for numerical computation. It immediately had great success in the Machine Learning community and in less than one year it also had a lot of support and development by Google itself, moreover by many community projects, developed in any area of Deep Learning. The peculiarity of TensorFlow is its work flow, made by data flow graphs.

Where Nodes represent mathematical operations, edges represent the multidimensional data arrays communicated between them; the latter can be considered, as in electronics, a Tensor, from here its name. In May 2016, Google **[56]** revealed that it has used TensorFlow in the AlphaGo project, with a special hardware dedicated to boost the library's performance. To rapidly reach this goal, Google dedicated a special attention to the user experience of TensorFlow , which arrives with a great basic support and a well grown GitHub community **[57]**, the key of this quick improvement. All these are the peculiarities that pushed us to choose TensorFlow, over more developed and bigger communities. The disruption made in the Deep Learning environment shows up its great future potentials that are rolling out day by day.

Google colaboratory

Figure 4.6: Colab logo.

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. Google Colab's major differentiator from Jupyter Notebook is that it is cloud-based and Jupyter is not. This means that if you work in Google Colab, you do not have to worry about downloading and installing anything to your hardware**[57]**

Machine specs

Since we are using google colab we can get plenty of performant hardware we cite below the free plan

CPU	Intel(R) Xeon(R) CPU @ 2.20GHz
RAM	13GB
Hard disk space	40GB
GPU	Tesla T4 16GB

Table 4.1: Machine specs.

2. Project description

In this project we aimed to construct a neural collaborative filtering recommender system with TensorFlow library, where recommendations of books are built upon the existing ratings of other users, who have similar ratings with the user to whom we want to recommend. This approach focuses on finding users who have given similar ratings to the same books, thus creating a link between users, to whom will recommend books that were reviewed in a positive way. In this way, we look for associations between users, not between books. Therefore, collaborative filtering relies only on observed user behavior to make recommendations — no profile data or content data is necessary.

3. Our work

In order to realize this work we passed by some important steps, starting by understanding the dataset, preparing our data to be ready for its role, creating the model and finally training the model to get results.

3.1 Dataset

In our work we are using the book-crossing dataset that can be found online **[58]**. To start we need to know the format of the data ,our dataset comprises 3 tables.

- **Users:** Contains the users' information. Note that user IDs ('User-ID') have been anonymized and mapped to integers. Demographic data is provided ('Location', 'Age') if available. Otherwise, these fields contain NULL-values.

- **Books:** Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (‘Book-Title’, ‘Book-Author’, ‘Year-Of-Publication’, ‘Publisher’), obtained from Amazon Web Services. Note that in case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavors (`Image-URL-S`, `Image-URL-M`, `Image-URL-L`), i.e., small, medium, large. These URLs point to the Amazon web site.
- **Book-Ratings:** Contains the book rating information. Ratings (`Book-Rating`) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0.

3.2 Data preparation

First we need to import some libraries, since we are using google colab most of the python libraries came preinstalled, so we just need to import them.

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
```

Next, reading dataset tables from our storage.

```
rating = pd.read_csv('data/BX-Book-Ratings.csv', sep=';', error_bad_lines=False,
encoding="latin-1")
user = pd.read_csv('data/BX-Users.csv', sep=';', error_bad_lines=False,
encoding="latin-1")
book = pd.read_csv('data/BX-Books.csv', sep=';', error_bad_lines=False,
encoding="latin-1")
```

After that we need to do some important steps to make the work clear, starting by Merging user, rating and book data and also removing unused columns.

```
book_rating = pd.merge(rating, book, on='ISBN')
cols = ['Year-Of-Publication', 'Publisher', 'Book-Author', 'Image-URL-S',
```

```
'Image-URL-M', 'Image-URL-L']
book_rating.drop(cols, axis=1, inplace=True)
```

Then Filtering books that have had at least 25 ratings, Filtering users that have given at least 20 ratings.

```
#count all ratings
rating_count = (book_rating.groupby(by = ['Book-Title'])['Book-Rating'].count()
    .reset_index().rename(columns = {'Book-Rating': 'RatingCount_book'})
    [['Book-Title', 'RatingCount_book']]
   )
#books that have had at least 25 ratings
threshold = 25
rating_count = rating_count.query('RatingCount_book >= @threshold')

user_rating = pd.merge(rating_count, book_rating, left_on='Book-Title',
right_on='Book-Title', how='left')
#counting users
user_count = (user_rating.
    groupby(by = ['User-ID'])['Book-Rating'].
    count().
    reset_index().
    rename(columns = {'Book-Rating': 'RatingCount_user'})
    [['User-ID', 'RatingCount_user']]
   )
# Users that have given at least 20 ratings
threshold = 20
user_count = user_count.query('RatingCount_user >= @threshold')

#Now let's combine users that gave rating to books
combined = user_rating.merge(user_count, left_on = 'User-ID',
     right_on = 'User-ID',
     how = 'inner')

print('Number of unique books: ', combined['Book-Title'].nunique())
print('Number of unique users: ', combined['User-ID'].nunique())
# output :
''''
   Number of unique books:  5850
   Number of unique users:  3192
''''
```

Our final data frame includes 3,192 users for 5,850 books. And every user gave at least 20 ratings and every book got at least 25 ratings.

Our technique will be based on the following observations:

- Users who rate books in a similar manner share one or more hidden preferences.
- Users with shared preferences are likely to give ratings in the same way to the same books.

Now we pass to the Process in TensorFlow in order to normalize the ratings feature, then build user-book matrix with three features:

```
scaler = MinMaxScaler()
combined['Book-Rating'] = combined['Book-Rating'].values.astype(float)
rating_scaled = pd.DataFrame(scaler.fit_transform(combined['Book-Rating']
                  .values.reshape(-1,1)))
combined['Book-Rating'] = rating_scaled

combined = combined.drop_duplicates(['User-ID', 'Book-Title'])
user_book_matrix = combined.pivot(index='User-ID', columns='Book-Title',
values='Book-Rating')
user_book_matrix.fillna(0, inplace=True)

users = user_book_matrix.index.tolist()
books = user_book_matrix.columns.tolist()
user_book_matrix = user_book_matrix.values
```

Since we are using TensorFlow 2.8.2, tf.placeholder is only available in v1, so we have to work around like so :

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

In the following code script we aim to

- Set the network parameters, such as the dimension of each hidden layer.
- Initialize the TF placeholder.
- Weights and biases are randomly initialized.

```
num_input = combined['Book-Title'].nunique()
num_hidden_1 = 10
num_hidden_2 = 5

X = tf.placeholder(tf.float64, [None, num_input])
```

```
weights = {
    'encoder_h1': tf.Variable(tf.random_normal([num_input, num_hidden_1],
dtype=tf.float64)),
    'encoder_h2': tf.Variable(tf.random_normal([num_hidden_1, num_hidden_2],
dtype=tf.float64)),
    'decoder_h1': tf.Variable(tf.random_normal([num_hidden_2, num_hidden_1],
dtype=tf.float64)),
    'decoder_h2': tf.Variable(tf.random_normal([num_hidden_1, num_input],
dtype=tf.float64)),
}

biases = {
    'encoder_b1': tf.Variable(tf.random_normal([num_hidden_1], dtype=tf.float64)),
    'encoder_b2': tf.Variable(tf.random_normal([num_hidden_2], dtype=tf.float64)),
    'decoder_b1': tf.Variable(tf.random_normal([num_hidden_1], dtype=tf.float64)),
    'decoder_b2': tf.Variable(tf.random_normal([num_input], dtype=tf.float64)),
}
```

3.3 The model

Now, we can build the encoder and decoder model, as follows:

```
def encoder(x):
   layer_1 = tf.nn.sigmoid(tf.add(tf.matmul(x, weights['encoder_h1']),
biases['encoder_b1']))
   layer_2 = tf.nn.sigmoid(tf.add(tf.matmul(layer_1, weights['encoder_h2']),
biases['encoder_b2']))
   return layer_2

def decoder(x):
   layer_1 = tf.nn.sigmoid(tf.add(tf.matmul(x, weights['decoder_h1']),
biases['decoder_b1']))
   layer_2 = tf.nn.sigmoid(tf.add(tf.matmul(layer_1, weights['decoder_h2']),
biases['decoder_b2']))
   return layer_2
```

Now, we construct the model and the predictions before we define loss function and optimizer, and minimize the squared error, and define the evaluation metrics

```
encoder_op = encoder(X)
decoder_op = decoder(encoder_op)
```

```
y_pred = decoder_op
y_true = X

loss = tf.losses.mean_squared_error(y_true, y_pred)
optimizer = tf.train.RMSPropOptimizer(0.03).minimize(loss)
eval_x = tf.placeholder(tf.int32, )
eval_y = tf.placeholder(tf.int32, )
pre, pre_op = tf.metrics.precision(labels=eval_x, predictions=eval_y)
```

Because TensorFlow uses computational graphs for its operations, placeholders and variables must be initialized before they have values. So in the following code, we initialize the variables, then create an empty dataframe to store the result table, which will be top 10 recommendations for every user.

```
init = tf.global_variables_initializer()
local_init = tf.local_variables_initializer()
pred_data = pd.DataFrame()
```

3.4 Training

Finally we start training our model, by splitting training data into batches then we feed our network. We train our model with vectors of user ratings, each vector represents a user and each column a book, and entries are ratings that the user gave to books. After a few trials, we've discovered that a training model for 100 epochs with a batch size of 35 would be consuming enough memories. This means that the entire training set will feed our neural network 100 times, every time using 35 users. At the end, we must make sure to remove user's ratings in the training set. In order to not recommend books to a user in which he has already rated.

```
with tf.Session() as session:
   epochs = 100
   batch_size = 35

   session.run(init)
   session.run(local_init)

   num_batches = int(user_book_matrix.shape[0] / batch_size)
   user_book_matrix = np.array_split(user_book_matrix, num_batches)
```

```
    for i in range(epochs):

        avg_cost = 0
        for batch in user_book_matrix:
            _, l = session.run([optimizer, loss], feed_dict={X: batch})
            avg_cost += l

        avg_cost /= num_batches

        print("epoch: {} Loss: {}".format(i + 1, avg_cost))

    user_book_matrix = np.concatenate(user_book_matrix, axis=0)

    preds = session.run(decoder_op, feed_dict={X: user_book_matrix})

    pred_data = pred_data.append(pd.DataFrame(preds))

    pred_data = pred_data.stack().reset_index(name='Book-Rating')
    pred_data.columns = ['User-ID', 'Book-Title', 'Book-Rating']
    pred_data['User-ID'] = pred_data['User-ID'].map(lambda value: users[value])
    pred_data['Book-Title'] = pred_data['Book-Title'].map(lambda value:
books[value])

    keys = ['User-ID', 'Book-Title']
    index_1 = pred_data.set_index(keys).index
    index_2 = combined.set_index(keys).index

    top_ten_ranked = pred_data[~index_1.isin(index_2)]
    top_ten_ranked = top_ten_ranked.sort_values(['User-ID', 'Book-Rating'],
ascending=[True, False])
    top_ten_ranked = top_ten_ranked.groupby('User-ID').head(10)
```

Fortunately at **epoch: 100**, we got **Loss = 0.002676744400370088** which is a good result

After the training let's see how our model works. Selecting a random user to see what books we should recommend to him.

```
top_ten_ranked.loc[top_ten_ranked['User-ID'] == 6543]
```

	User-ID	Book-Title	Book-Rating
294337	6543	Harry Potter and the Chamber of Secrets (Book 2)	0.045886
294341	6543	Harry Potter and the Prisoner of Azkaban (Book 3)	0.043756
294343	6543	Harry Potter and the Sorcerer's Stone (Harry P...	0.039233
294339	6543	Harry Potter and the Goblet of Fire (Book 4)	0.037726
297901	6543	To Kill a Mockingbird	0.037326
292857	6543	Angels & Demons	0.035808
294340	6543	Harry Potter and the Order of the Phoenix (Boo...	0.035007
298190	6543	Where the Heart Is (Oprah's Book Club (Paperba...	0.033946
293725	6543	Divine Secrets of the Ya-Ya Sisterhood: A Novel	0.031665
292683	6543	A Time to Kill	0.031400

Figure 4.7: Top 10 books recommended to user ‘6543’.

General conclusion

The work presented in this thesis falls within the framework of the exploitation of deep learning techniques in recommender systems. Specifically, we implemented and evaluated a collaborative filtering algorithm, based on neural networks (NCF) .In our case, we could see that the NCF algorithm gave very acceptable results. We succeeded, through the series of tests carried out on the algorithm, in observing the impact of hyperparameters on the performance of the model. The evaluation made using the metrics allowed us to deduce that the neural network technique presents the best results for our dataset.

Indeed, we were confronted with various difficulties, for example at the level of the implementation of the algorithm, however it was a real experience and a chance for the discovery of the field of deep learning. As perspectives on this work, it would be interesting to do this study on different datasets to show if the choice of the algorithm depends on the dataset, its size and also on the hyperparameters of the algorithm.

One of the things that we can use to directly improve our model, could be using a more specialized dataset, acquiring such a dataset will be a challenge in itself, one way would be to start a community project to build a test dataset for the university library, so that researchers and students can use it to test new methods or be the end user of the system after building and deploying this project which will be a website available for all readers that recommend books for these last ,to improve their productivity in research and study area.

Bibliography

[1] Tom M. Mitchell. Machine Learning. MGH, 1997. isbn: 9780070428072; 0070428077. url: http://www.cs.cmu.edu/~tom/mlbook.html.

[2] Jose Galindo. Handbook of Research on Fuzzy Information Processing in Databases. 1st ed. 2008. isbn: 1599048531; 9781599048536; 9781599048543; 159904854X. url: https://www.igi- global.com/book/handbook- research- fuzzy- informationprocessing/469.

[3] Ilias G. Maglogiannis, Emerging Artificial Intelligence Applications in Computer Engineering ,2007, isbn:1586037803,9781586037802,9781435625198.

[4] Haim Dahan, Shahar Cohen, Lior Rokach, Proactive Data Mining with Decision Trees,Springer-Verlag New York 2014, isbn: 9781493905386, 9781493905393.

[5] Lambert M. Surhone, Miriam T. Timpledon, Susan F. Marseken,Naive Bayes Classifier: Classifier (mathematics), Bayes' Theorem, Betascript Publishing, 2010, isbn:613033446X, 9786130334468.

[6] Julian Avila, Scikit-Learn Cookbook: Over 80 Recipes for Machine Learning in Python With Scikit-Learn, Packt Publishing, 2017, isbn: 9781787286382.

[7] Trevor Hastie, Robert Tibshirani, Jerome H. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer, 2009, isbn : 0387848843, 9780387848846.

[8] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541-551.

[9] Fukushima, Kunihiko. (2021). Artificial Vision by Deep CNN Neocognitron. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 51. 76-90. 10.1109/TSMC.2020.3042785.

[10] Wikiversité La communauté pédagogique libre website, june 2022, url: https://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Allez_plus_loin_:_le_deep_Learning .

[11] Khelil, Mohamed Imed, and Mohamed Ladjal. "Hybrid Predictive Models for Water Quality Assessment Based on Water Quality Index Using ANN, LSSVM and multivariate statistical Methods." 9th (Online) International Conference on Applied Analysis and Mathematical Modeling (ICAAMM21) June 11-13, 2021, Istanbul-Turkey .

[12] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics 5: 115-133, 1943.

[13] D. Hebb, The Organization of Behavior, New York: Wiley, 1949.

[14] B. Widrow and M. Hoff, "Adaptive switching circuits," 1960 IRE WESCON Convention Record, New York: IRE, pp. 96-104, 1960.

[15] Kohonen, Teuvo. (2007). Description of Input Patterns by Linear Mixtures of SOM Models.

[16] Hopfield, John. (1984). Neurons With Graded Response Have Collective Computational Properties Like Those of Two-State Neurons. Proceedings of the National Academy of Sciences of the United States of America. 81. 3088-92. 10.1073/pnas.81.10.3088.

[17] Toulouse University url:www.math.univ-toulouse.fr/~besse/Wikistat/pdf/stm-app-rn.pdf

[18] Korean J Radiol. 2017 Jul-Aug;18(4):570-584. https://doi.org/10.3348/kjr.2017.18.4.570.

[19] Heaton, J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genet Program Evolvable Mach 19, 305–307 (2018). https://doi.org/10.1007/s10710-017-9314-z .

[20] Szandała, Tomasz. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks.

[21] Indian tech warrior, june 2022, url: https://indiantechwarrior.com/7-types-of-neural-network-activation-functions-how-to-choose

[22] Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, Ptrick Haffner, Proc of the IEEE nov 1998.

[24] Gunay Abdullayeva et al. "Application and Evaluation of LSTM Architectures for Energy Time-Series Forecasting Master's Thesis (30 ECTS) Application and Evaluation of LSTM Architectures for Energy Time-Series Fore- casting". PhD thesis. May 2019. doi: 10.13140/RG.2.2.20433.76641.

[25] dataanalytics(website), june 2022, url : https://dataanalyticspost.com/Lexique/auto-enco

[26] (website), june 2022 url: https://community.canvaslms.com/t5/Canvas-Developers-Group/Canvas-LMS-Cheat-Detection-System-In-Python/m-p/118134 .

[27] Lebigdata website, june 2022,url: https://www.lebigdata.fr/reinforcement-learning-definition

[28] G. Adomavicius and A. Tuzhilin. “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions”. In: IEEE Transactions on Knowledge and Data Engineering 17.6 (2005), pp. 734–749. doi: 10.1109/TKDE. 2005.99.

[29] David Goldberg et al. “Using Collaborative Filtering to Weave an Information Tapestry”. In: Commun. ACM 35.12 (Dec. 1992), pp. 61–70. issn: 0001-0782. doi: 10.1145/ 138859.138867. url: https://doi.org/10.1145/138859.138867.

[30] Paul Resnick and Hal R. Varian. “Recommender Systems”. In: Commun. ACM 40.3 (Mar. 1997), pp. 56–58. issn: 0001-0782. doi: 10.1145/245108.245121. url: https: //doi.org/10.1145/245108.245121.

[31] Upendra Shardanand and Pattie Maes. “Social Information Filtering: Algorithms for Automating “Word of Mouth””. In: CHI ’95 (1995), pp. 210–217. doi: 10.1145/ 223904.223931. url: https://doi.org/10.1145/223904.223931.

[32] Robin Burke. “Hybrid Recommender Systems: Survey and Experiments”. In: User Modeling and User-Adapted Interaction 12.4 (Nov. 2002), pp. 331–370. issn: 0924- 1868. doi: 10 . 1023 / A : 1021240730564. url: https://doi.org/10.1023/A:1021240730564.

[33] An-Te Nguyen. "COCOFIL2 : Un nouveau système de filtrage collaboratif basé sur le modèle des espaces de communautés". Thèses. Université Joseph-Fourier - Grenoble I, Nov. 2006. url: https://tel.archives-ouvertes.fr/tel-00353945.

[34] An-Te Nguyen, Nathalie Denos, and Catherine Berrut. “Cartes de communautés pour l’adaptation interactive de profils dans un système de filtrage d’information”. In: INFORSIDE. 2005.

[35] Francesco Ricci. “Travel Recommender Systems”. In: IEEE Intelligent Systems (Nov. 2002), pp. 55–57.

[36] K. Jothi et al. “Qualitative Analysis of Models and Issues in Recommender Systems”. In: Journal of Computational and Theoretical Nanoscience 16 (May 2019), pp. 1881– 1888. doi: 10.1166/jctn.2019.7819.

[37] C.J Van Rijsbergen. 2nd ed. London, Boston: Butterworths, 1979.

[38] (website), June 2022, url: www.r-bloggers.com/2016/07/recosystem-recommender-system-using-parallel-matrixfactorization/.

[39] Shaikhah Alotaibi and Julita Vassileva. “Effect of Different Implicit Social Networks on Recommending Research Papers”. In: Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization. UMAP ’16. Halifax, Nova Scotia, Canada: Association for Computing Machinery, 2016, pp. 217–221. isbn: 9781450343688. doi: 10.1145/2930238.2930293. url: https://doi.org/10.1145/2930238.2930293.

[40] Xiaoyuan Su and Taghi M. Khoshgoftaar. “A Survey of Collaborative Filtering Techniques”. In: Adv. in Artif. Intell. 2009 (Jan. 2009). issn: 1687-7470. doi: 10.1155/ 2009/421425. url: https://doi.org/10.1155/2009/421425.

[41] Robin Burke. “Hybrid Recommender Systems: Survey and Experiments”. In: User Modeling and User-Adapted Interaction 12.4 (Nov. 2002), pp. 331–370. issn: 0924- 1868. doi: 10 . 1023 / A : 1021240730564. url: https : / / doi . org / 10 . 1023 / A : 1021240730564.

[42] Fidel Cacheda et al. “Comparison of Collaborative Filtering Algorithms: Limitations of Current Techniques and Proposals for Scalable, High-Performance Recommender Systems”. In: ACM Trans. Web 5.1 (Feb. 2011). issn: 1559-1131. doi: 10.1145/ 1921591.1921593. url: https://doi.org/10.1145/1921591.1921593.

[43] (website), june 2022, url: https://nealanalytics.com/blog/how-to-build-a-personalized-recommender-using-datascience/

[44] (website), june 2022, url: https://python.plainenglish.io/recommendation-systems-content-based-recommendationd8f9da5bfac.

[45] Pinnapareddy, Nishanth Reddy, "Deep Learning based Recommendation Systems" (2018). Master's Projects. 644. DOI: https://doi.org/10.31979/etd.k9vm-pxcs https://scholarworks.sjsu.edu/etd_projects/644.

[46] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme {srendle, freudenthaler, gantner, schmidt-thieme}@ismll.de Machine Learning Lab, University of Hildesheim Marienburger Platz 22, 31141 Hildesheim, Germany.

[47] (website), june 2022, url: https://www.jeremyjordan.me/autoencoders.

[48] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web (WWW '17). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569.

[49] Thomas Schmitt, Collaborative matching of job offers and requests in Orsay, 06/29/2018, url: https://tel.archives-ouvertes.fr/tel-01886623/document.

[50] Python. What is Python? June 1, 2022. url: https://www.python.org/doc/essays/blurb/

[51] "Numpy official website," [Online]. Available: https://www.numpy.org/ .

[52] Pandas official website, https://pandas.pydata.org/docs/.

[53] SciKit learn, June 2022, url: https://scikit-learn.org/stable/index.html

[54] Google. Alphabet. June., 2022. url: https://abc.xyz/.

[55] Google. TensorFlow. June. 2022. url: https://www.tensorflow.org/

[56] The Verge. Google revelead AlphaGo secret Hardware with ASIC chip-tensor Processor Unit for Machine-Learning. june 2022. url: http://www.theverge. com/circuitbreaker/2016/5/19/11716818/google-alphago-hardwareasicchip-tensorprocessor-unit-machine-learning.

[57] Google. Tensorflow GitHub repository. une. 2022. url: https://github. com/tensorflow.

[58] Google Colab official website, url: https://colab.research.google.com/

[59] Book Crossing Dataset, May 2022, url:http://www2.informatik.uni-freiburg.de/~cziegler/BX/

Abstract

In recent years deep learning techniques have shown great results in many fields (such as natural language processing, computer vision). Among their new applications, researchers started to apply these techniques on recommender systems to predict the best user preferred choice. In this project we applied one of deep learning techniques which is neural networks, specifically NCF on bookCrossing dataset to recommend books to users based on their previous ratings. Our model showed good results after extensive evaluation and experimental tests.

Keywords: deep learning, recommender systems, TensorFlow, neural collaborative filtering.

Résumé

Ces dernières années, les techniques d'apprentissage en profondeur ont montré d'excellents résultats dans de nombreux domaines (tels que NLP, la vision par ordinateur). Parmi leurs nouvelles applications, les chercheurs ont commencé à appliquer ces techniques sur des systèmes de recommandation pour prédire le meilleur choix préféré de l'utilisateur. Dans ce projet, nous avons appliqué l'une des techniques d'apprentissage en profondeur qui est les réseaux de neurones, en particulier NCF sur l'ensemble de données bookCrossing pour recommander des livres aux utilisateurs en fonction de leurs évaluations précédentes. Notre modèle a montré de bons résultats après une évaluation approfondie et des tests expérimentaux.

Mots clés: apprentissage profond, systèmes de recommandation, TensorFlow, filtrage collaboratif neuronal.

ملخص

في السنوات الأخيرة ، أظهرت تقنيات التعلم العميق نتائج رائعة في العديد من المجالات (مثل معالجة اللغة الطبيعية ورؤية الكمبيوتر). بدأ الباحثون في تطبيق هذه التقنيات على أنظمة التوصية للتنبؤ بالاختيار الأفضل للمستخدم. في هذا المشروع قمنا بتطبيق إحدى تقنيات التعلم العميق وهي الشبكات العصبية ، وتحديداً NCF على مجموعة بيانات bookCrossing للتوصية بالكتب للمستخدمين بناءً على تقييماتهم السابقة. أظهر نموذجنا نتائج جيدة بعد التقييم الشامل والاختبارات التجريبية.

كلمات مفتاحية:التعلم العميق ، أنظمة التوصية ، TensorFlow ، التصفية التعاونية العصبية.