REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Logiciel

Par :

CHERFI Boutheina

Sur le thème

Sequential Recommendation Using Convolutional Neural Networks

Soutenu publiquement le 26 / 09 / 2021 à Tiaret devant le jury composé de :

Mr DAOUD Mohamed Amine	Grade	Maître assistant	Président
Mr BOUDAA Boudjemaa	Grade	Maître de conférences	Encadreur
Mme LAARADJ Zohra	Grade	Maître assistant	Examinateur

2020-2021

Abstract

With the prevalence of information technology (IT), recommender system has long been acknowledged as an effective tool for addressing information overload problem, which makes users easily filter and locate information of their preferences, and allows online platforms to widely publicize the information they produce. In the field of sequential recommendation, deep learning methods have received a lot of attention in the past few years and surpassed traditional models such as Markov chain-based and factorization-based ones. In this view, this thesis focuses on DL-based sequential recommender systems by taking the aforementioned issues into consideration. Specifically, we illustrate the concept of sequential recommendation, propose a convolutional neural network model to show the effectiveness of sequential recommenders based on CNNs in several real-life scenarios.

Keywords: Recommender systems, Sequential Recommendation, Sequential data, Neural networks, Convolutional neural networks, Next-Item recommendation.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Mr. Boudaa Boudjemaa, whose sincerity and encouragement I will never forget. He has been an inspiration as I hurdled through the path of this Masters degree. He is the true definition of a leader and the ultimate role model. This thesis would not have been possible without him, whose guidance from the initial step in research enabled me to develop an understanding of the subject. I am thankful for the extraordinary experiences he arranged for me and for providing opportunities for me to grow professionally. It is an honor to learn from Mr. Boudaa Boudjemaa.

Thank you to my sister, Hafsa, for always being there for me and for telling me that I am awesome even when I didn't feel that way. Thank you for being my best friend.

CHERFI Boutheina

Dedication

"For everyone who did not have the same opportunities."

Contents

List of Figures

List of Tables

Acronyms

RS	**R**ecommender **S**ystem
SRS	**S**equential **R**ecommender **S**ystem
CBRS	**C**ontent **B**ased **R**ecommender **S**ystem
CF	**C**ollaborative **F**iltering
SARS	**S**equence **A**ware **R**ecommender **S**ystem
CNN	**C**onvolutional **N**eural **N**etwork
GNN	**G**raph **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
GRU	**G**ated **R**ecurrent **U**nit
DL	**D**eep **L**earning
AI	**A**rtificial **I**ntelligence
ML	**M**achine **L**earning
NLP	**N**atural **L**anguage **P**rocessing
DNN	**D**eep **N**eural **N**etworks
CRM	**C**ustomer **R**elationship **M**anagement
ANN	**A**rtificial **N**eural **N**etworks
MLP	**M**ulti **L**ayer **P**erceptron
BPTT	**B**ack**p**ropagation **T**hrough **T**ime
LSTM	**L**ong **S**hort **T**erm **M**emory
GD	**G**radient **D**escent

Introduction

A. Background

In our everyday life we make decisions countless times, sometimes even unnoticed. How to dress up in the morning considering our schedule? Which menu to pick in the restaurant? Which television to buy? Thousands of these everyday questions have to be answered besides couple of crucial ones day by day. To make the correct decision in some cases we ask our friend's opinion or an expert, but there are other options that raised recently to support us in our decisions. We not necessarily need to nag the librarian or the vendor at the bookstore finding a book that would fit to our mood and preferences, if we have the possibility using Amazon to recommend us a book based on our previous readings. Youtube also recommends audiovisual content to its users based on their previous views.

It is the recommender system which is considered one among the most powerful tools in the present digital world. Explanations are usually provided by it to their recommendations so that web users are helped to find its products, people and also their friends who are missing in social communities. In the field of recommender system, there are various methods and approaches which have been implemented.

Over the last two decades, there has been much work in both industry and academia to develop new approaches to recommender systems. The interest still remains high because it constitutes a problem-rich research area and because of the abundance of practical applications that help users deal with information overload and provide personalised recommendations, content, and services to them.

One of the first and quite successful research on this subject was the Music Genome Project in 1999, which aims to "understand" and capture music through its properties. To this end, more than 450 such properties have been revealed and their relations have been described using an algorithm. The basis of the procedure is that when the user likes a song, positive values are assigned to its specific properties (such as style, era, artist, orchestration, beat, etc.). Songs with similar properties will then be further promoted on the preference list and brought to the user's attention. The huge advantage to collaborative filtering is that very little information is sufficient at startup, while the former unfortunately requires a lot of users and feedback to identify people with similar tastes. However, the disadvantage is that usually it can hardly, or not at all make recommendations outside the user's music lists, since it is not building on the similarities between users, only the "understanding" the properties of music as an entity. Pandora Internet Radio, which has 250 million users, is based on this project even today [1].

We have learned that people are hungry for effective tools for information filtering, and that collaborative filtering is an exciting complement to existing filtering systems. Users value both the taste-based recommendations, and the sense of community they get by participating in a group filtering process. However, there are many open research problems still in collaborative filtering, In the other hand there is an increasing attention on sequential recommendation systems to infer the dynamic user preferences with sequential user interactions. While the semantics of an item can change over time and across users, the item correlations defined by user interactions in the short term can be distilled to capture such change, and help in uncovering the dynamic user preferences.

The recommendation system is based on recommended techniques. The recommended technique, also known as personalized information filtering, is used to predict whether a given user will like a particular project (predictive problem) or to identify a set of N items of interest to a given user (top-N recommendation) problem). The recommendation system proactively provides users with items that may be of interest, essentially by linking users and projects in a certain way, the input data source is followed by a recommendation algorithm to generate recommendation results for personalized recommendation. Different recommendation systems use different recommendation algorithms, so the core of the recommendation system is to use different recommendation algorithms according to different data sources [2].

Recently, deep learning has been revolutionizing the recommendation architectures dramatically and brings more opportunities to improve the performance of recommender. Recent advances in deep learning based recommender systems have gained significant attention by overcoming obstacles of conventional models and achieving high recommendation quality [3].

Given the practical importance of recommender systems, there has been increased attention to sequential recommender systems in recent years. These systems differ from many legacy recommenders both in terms of the input they process and the way they consider sequential information when generating the recommendations. Regarding the input, sequential recommenders assume that the available preference information is chronologically ordered or contains timestamps. As for the outputs, sequential methods take this ordering into account when generating the recommendations, e.g., by focusing on recent data, making repeated recommendations, or identifying temporal dependencies in the data.

B. Problem Statement

Browsing on an e-commerce site, is naturally a sequential task, therefore using a matrix completion formulation would neglect the sequential nature of the data. Using sequential user interactions is especially important when there are many new or anonymous users as no long-term historical data about the general tastes of these users is available. Although convolutional neural networks have shown good results in sequence modelling tasks in other domains, investigating their usage in sequential recommender systems remains an open topic of research. The main research question of this thesis is thus formulated as:

How can convolutional neural networks

be used in sequential recommender

systems?

C. Delimitations

This study is to focus on sequential recommender systems, Data used by sequential recommender systems called implicit feedback, keeps the sequential order of actions. It is in this context that sequential recommender systems have emerged.

However, e-commerce companies are not the only ones that use recommendation systems to push customers to buy additional products. There are use cases in entertainment, gaming, education, advertising, home decor, and some other industries. There are different applications, from recommending music and events to furniture and dating profiles.

D. Approach

To answer the research question, a sequential recommender system is presented, Our deep learning approach is inspired by recent text generation techniques that use CNN models capable of generating subsequent words from a given phrase (sequence of words). For our recommendation problem, we replaced sequence of words with sequence of items and applied the same methodology.

E. Outline

This thesis is structured in three chapters besides a general introduction and a general conclusion:

- ☐ **INTRODUCTION**

  An initiation to recommender systems and the background, problem statement, and delimitations of the thesis.

- ☐ **CHAPTER ONE: SEQUENTIAL RECOMMENDER SYSTEMS**

  In the 1st chapter we present a brief summary on the types of recommender systems and a general overview of sequential recommender systems and approaches used for.

- ☐ **CHAPTER TWO: CONVOLUTIONAL NEURAL NETWORKS**

  We dedicate the 2nd chapter to the convolutional neurale networks structure, types and process.

- ☐ **CHAPTER THREE: SEQUENTIAL RECOMMENDER SYSTEM USING CONVOLUTIONAL NEURAL NETWORKS**

  Using concepts from the 1st and 2nd chapter, the next chapter introduce our approach that use convolutional neural networks for a sequential recommender system development.

- ☐ **CONCLUSION**

  The results of this thesis are summarised in the conclusion where we also designate the directions of our future research.

Chapter 1

Sequential Recommender Systems

"You might also like ... to know how programs know what you want before you do."

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user [4].

1.1 Introduction

The internet and modern web services have been increasing within the last few decades; a surplus of information is now accessible to everyone. It can be challenging for users to filter through all this information and take away essential aspects [5]. Recommender Systems (RSs) have evolved into a fundamental tool for making more informative, efficient and effective choices and decisions in almost every daily aspect of life, working, business operations, study, entertaining and socialization [6]. A lot of RS models and techniques have been proposed, including content-based RSs and collaborative filtering-based RSs but they can only model the user-item interactions in a static way and capture the users' general preferences. In contrast, SRSs model the sequential user behaviors, the interactions between users and items, and the evolution of user's preferences and item popularity over time to capture the current and recent preference of a user. A sequential recommendation system (SRS) aims to predict if an item would be useful to a user based on his historical information. The use of these systems has been steadily growing within the last few years. SRSs involve the above aspects for more accurate, customized and dynamic recommendations.

In this chapter, we present a comprehensive guide to RSs. We highlight different RSs categories and a methodological overview of the sequence-aware recommender systems is given.

1.2 Recommender Systems

Recommender system is defined as a decision making strategy for users under complex information environments [7]. Recommender systems are information filtering systems that deal with the problem of information overload by filtering vital information fragment out of large amount of dynamically generated information according to user's preferences, interest, or observed behavior about item [8].

1.3 Data used by recommender systems

Data used by RSs refers to three kinds of objects: items, users, and transactions, i.e., relations between users and items [9].

Items. Items are the objects that are recommended. Items may be characterized by their complexity and their value or utility. The value of an item may be positive if the item is useful for the user, or negative if the item is not appropriate and the user made a wrong decision when selecting it.

Users. Users of a RS, may have very diverse goals and characteristics. In order to personalize the recommendations and the human-computer interaction, RSs exploit a range of information about the users. This information can be structured in various ways and again the selection of what information to model depends on the recommendation technique.

Transactions. We generically refer to a transaction as a recorded interaction between a user and the RS. Transactions are log-like data that store important information generated during the human-computer interaction and which are useful for the recommendation generation algorithm that the system is using.

1.4 How do we provide data for recommender systems ?

Called information collection phase This phase gathers vital information about users and prepares user profiles based on the users' attribute, behaviors, or resources accessed. Without constructing a well-defined user profile, the recommendation engine cannot work properly. A recommendation system is based on inputs that are collected in different ways, such as explicit feedback, implicit feedback, and hybrid feedback [10].

Explicit and implicit feedback provides different degrees of expressivity of the user's preferences [11]:

1.4.1 Explicit feedback

In explicit feedback , the user will provide ratings for items on a Likert scale. The rating scale will usually go from 'I like it a lot' to 'I do not like it'. Thus explicit feedback captures both positive and negative user preferences.

1.4.2 Implicit feedback

Implicit feedback can only be positive. For example, if a user did not listen to a track that does not imply he does not like the track.

1.4.3 Hybrid feedback

The combination of both explicit and implicit feedback is known as hybrid feedback.

Explicit feedback is generally more accurate than implicit feedback in representing the user's interests (although this is dependent on the domain and the RS application) [11].

In fact, ratings are the most popular form of transaction data that a RS collects.These ratings may be collected explicitly or implicitly. In the explicit collection of ratings, the user is asked to provide his opinion about an item on a rating scale. Ratings can take on a variety of forms [9]:

Numerical ratings such as the 1-5 stars provided in the book recommender associated with Amazon.com.

Ordinal ratings, such as "strongly agree, agree, neutral, disagree, strongly disagree" where the user is asked to select the term that best indicates her opinion regarding an item (usually via questionnaire).

Binary ratings that model choices in which the user is simply asked to decide if a certain item is good or bad.

Unary ratings can indicate that a user has observed or purchased an item, or otherwise rated the item positively.

In such cases, the absence of a rating indicates that we have no information relating the user to the item (perhaps he purchased the item somewhere else).

1.5 How does a recommender systems work?

Recommender systems are machine learning systems that help users discover new product and services. Every time you shop online, a recommendation system is guiding you towards the most likely product you might purchase. Recommender systems function with two kinds of information:

Characteristic information: This is information about items (keywords, categories, etc.) and users (preferences, profiles, etc.).

User-item interactions: This is information such as ratings, number of purchases, likes, etc.

1.6 Goals of Recommender Systems

From a business point of view, the primary goal of a recommender system is of course to increase the profit of the company, through a higher number of sales. To reach this objective, the system has to be able to meet some requirements [12]:

Know what the user wants / Relevance. quite obviously, the RS has to suggest items that are relevant to the users' tastes, because users are more likely to buy or consume items they have interest in.

Diversity, Novelty and Serendipity. firstly, if the RS always suggests items of the same sort to a user, there is a risk that this user would get bored or would not like any item. Thus, the system has to pay attention to put items of different types, bringing diversity.

Secondly, even if it can be relevant, the recommended item also has to be something the user has not already bought or experienced in the past: some novelty is required.

Lastly, the serendipity implies to surprise the user by recommending items he does not expect. Compared to novelty, the suggested item would belong to a category the user did not expect at all. This can sometimes lead the user to widen its area of interest, and help to increase sales diversity.

User satisfaction and fidelity. finally, another goal of the RS is to increase the user satisfaction and fidelity. A good user interface and accurate recommendations might encourage the user to connect and use the site again.

These key goals are common to all recommender systems in any application case, and they need to be integrated carefully during the implementation of the system.

1.7 Recommender Systems Types

There are majorly six types of recommender systems which work primarily in the Media and Entertainment industry, they are classifed in [9, 12, 13] into the following categories:

1.7.1 Collaborative-Filtering Recommendation Systems

Collaborative Filtering Recommendation System recommends to the active user the items that other users with similar tastes liked in the past. The similarity in taste of two users is calculated based on the similarity in the rating history of the users. Collaborative filtering techniques are classified into item-based filtering and user-based filtering [9].

1.7.2 Content-Based Recommendation Systems

Content-based approaches use external information about the items, such as keywords, tags, or profile written as texts in actual language to perform recommendation. By using item features, the system can provide recommendations of non-experienced items that have a similar thematic to those he liked in the past [12]. Measuring the utility of content-based filtering is commonly calculated by using heuristic functions, such as the cosine similarity metric. Content-based filtering can be employed in many cases, where the features' values can easily be extracted [13].

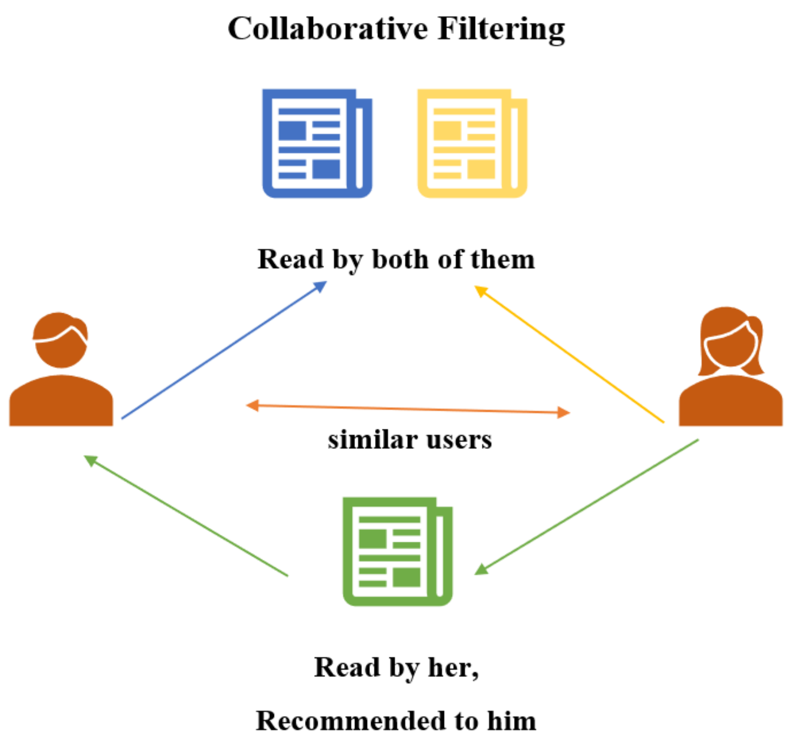

Figure 1.1: Collaborative-Filtering Recommendation System.

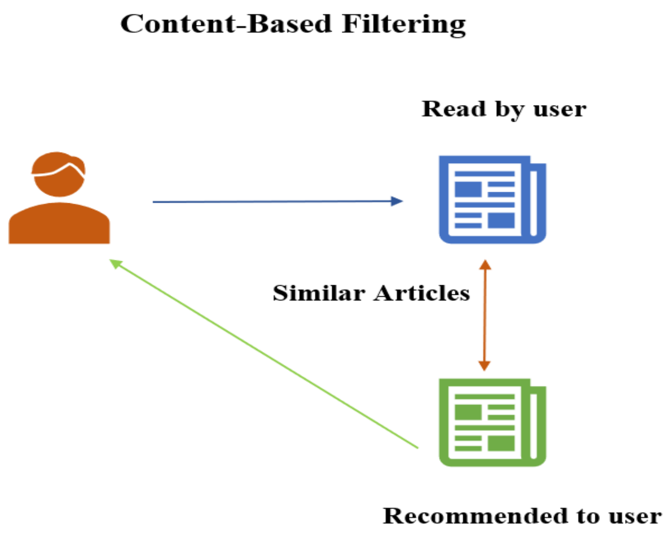

Figure 1.2: Content-Based Recommendation System.

1.7.3 Hybrid-Based Recommendation Systems

Hybrid systems are combining two or more techniques to obtain better performance [13]. Hybrid RS make the assumption that various sources of input are available at the same time, which allow to make use of different recommendation approaches inside one recommendation framework, and combine them to minimize their disadvantages [12].

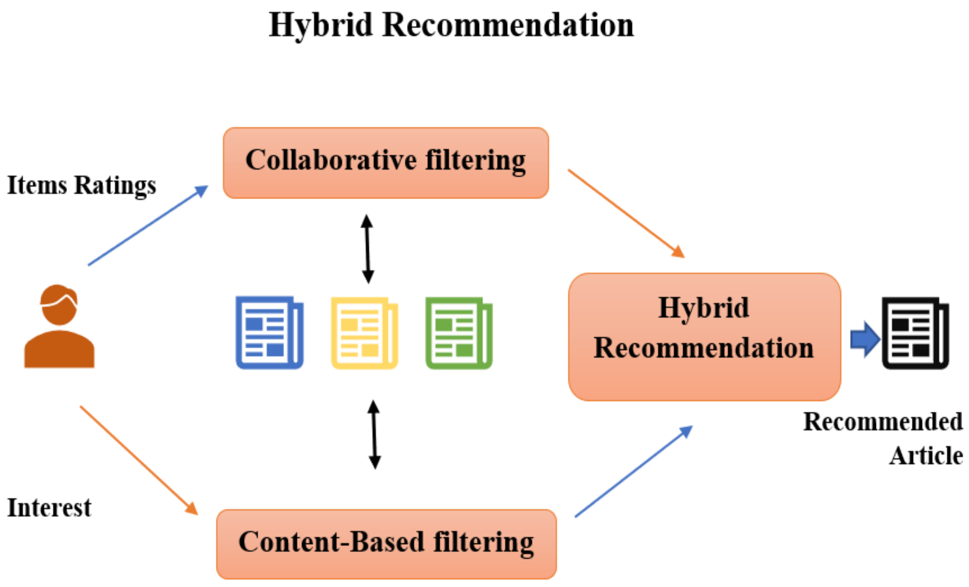

Figure 1.3: Hybrid RS combine Content-Based and Collaborative filtering .

1.7.4 Demographic-Based Recommendation Systems

This type of systems assumes the possibility of partitioning the set of users based on their demographic profile. The demographic features such as the country or age of each user will decide to which class he belongs to. Then, a set of rules decides which recommendation to perform depending on the class to which the user belongs [12]. Demographic RSs are especially useful when the amount of product information is limited [13].

1.7.5 Utility-Based Recommendation Systems

Utility-based RS provides recommendations based on generating a utility model of each item for the user. This system builds multi-attribute users' utility functions and recommends the highest utility item based on each item's calculated user-utility explicitly. Utility-based RSs are useful because they can factor non-product attributes into utility functions, such as product availability and vendor reliability. They generate utility computation, which allows them to check both real-time inventory and features of an item. It enables the visualization of its status to the user [13].

1.7.6 Knowledge-Based Recommendation Systems

Knowledge-based systems recommend items based on specific domain knowledge about how certain item features meet users needs and preferences and, ultimately, how the item is useful for the user. Knowledge-based RSs are noted to be advantageous for several purposes [9].

1.8 Business Adoption and Applications

RSs were once a novelty technique used by very few e-commerce sites. Now, they have transformed into a serious tool that is drastically shaping the e-commerce world. Many of the largest e-commerce businesses utilize RSs to help users determine what they want, alleviating the information-overload problem. However, RSs are not limited to marketing products; they have been widely developed in the service industry. They can provide recommendations in many different areas ranging from location-based information to movies, music, images, books, etc [13].

Therefore, in this section, we classify RSs based on their business adoption into five categories of e-commerce, transportation, agriculture, healthcare, and media [13].

1.8.1 Recommendation Systems in e-Commerce

RSs are aimed at providing customized recommendations of products to customers of websites. They learn from the customer and recommend relevant products to the user. These systems personalize the experience while attaining user interest. RSs can enhance sales of e-commerce sites in three ways.

Browsers into buyers: Visitors to an e-commerce site often look over products without buying anything, but if a site displays relevant recommendations to a user, they are more likely to purchase.

Cross-sell: Recommendation techniques suggest additional products to the users, apart from the one they are already buying. With this, the average order size should increase over time

Loyalty: In an era where a competitor's site can be visited by a mere click or two, loyalty is essential. RSs personalize the site for each user, which builds the user-site relationship. The more a customer uses a system, the more they are training the system, the more loyal a customer becomes, which also improves the quality of recommendations, over time

1.8.2 Recommendation Systems in Transportation

RSs can assist in diverse ways with the increasing use of Global Positioning System (GPS)-enabled devices, especially mobile devices. Because information overload problems become worse when using mobile devices. The development of wireless communication services and position detection techniques such as RFID or GPS have promoted location-based information systems. RSs play a significant role in path recommendation, smart transport application of goods, tourism industry , or venue recommendation.We have categorized applications of RSs in transportation into:

- Recommending trip
- Recommending path
- Recommending popular activities in a location

- Recommending popular locations (restaurants)
- Recommending transporters (goods transporter, bus lines, drivers)

1.8.3 Recommendation Systems in the e-Health Domain

E-health and medical decisions are considered for RSs' research, aiming to help medical professionals take fast and proper medical decisions.

1.8.4 Recommendation Systems in Agriculture

In Agriculture, RSs have a significant impact on managing and using the resources efficiently, such as fertilizers, agrochemicals, irrigation.

1.8.5 Recommendation Systems in Media and Beyond

The technological developments and changes in media and the increasing number of people visiting cultural places have led to an increase in various cultural items and offers. Therefore, visitors are bombarded with the information, making it difficult for them to find their interests. Thus, recommendation systems have become a vital tool to provide suggestions that ease the information overload in this area.

1.9 The Impact of Recommender Systems

- 35% of the purchases on Amazon are the result of their recommender system, according to McKinsey.
- Recommendations are responsible for 70% of the time people spend watching videos on YouTube.
- 75% of what people are watching on Netflix comes from recommendations, according to McKinsey.
- Employing a recommender system enables Netflix to save around $1 billion each year.

1.10 Problems Associated With Recommender Systems

Most of the conventional RSs, discussed previously, suffer from some serious drawbacks which restrain the effectiveness of the RSs. In this section, some of the major issues are discussed briefly [14]:

1.10.1 Limited content analysis

In CBRS, the accuracy of recommendation depends on the extent of user input provided. If the RS does not contain sufficient information about a user, the performance of the recommendation will be low. No CBR system can provide suitable suggestions if the analysed content does not contain enough information to discriminate items the user likes from items the user does not like. This problem is known as limited content analysis problem.

1.10.2 Over-specialisation

The aim of a RS is to help users explore new products. Diversity is an important feature of a good RS. Unfortunately, some recommendation algorithms may do exactly the opposite. They tend to recommend the popular and highly rated items which are liked by a particular user. This leads to lower accuracy as CBRS does not recommend items from a non-homogenous set of items. This is known as the over-specialisation problem.

1.10.3 Cold start

When a new item or a new user is introduced to an RS, the system will not have any past records (ratings, preferences, search history, etc.) on the basis of which recommendation should be made. This is known as the cold start problem. It is also termed as the new user problem or new item problem.

1.10.4 Sparsity

In practice, the RSs work with very large datasets. Hence, the user-item matrix used for CF is extremely sparse, which adversely affects the performances of the predictions or recommendations of the CF systems. It also takes place when a user, having used some particular product, did not bother to rate it. In other cases, users do not rate items that are not known to them.

1.10.5 Scalability

As the RSs work on large datasets, the complexity of the RSs increases in case of a huge number of users and millions of distinct items set. Many systems need to react immediately to online requirements and make recommendations for all users based on their purchases and rating history, which demands high scalability items.

1.10.6 Synonymy

Synonymy refers to the problem of multiple words having similar meanings. Most of the RSs are unable to find the same or similar items with different names (synonyms).

On account of this incapability, some associated problems emerge. For example, 'children movie' and 'children film' basically denote the same items, but memory-based CF systems would find no match between them to compute similarity.

1.10.7 Abbreviation

If the RS is not familiar with the abbreviations that the users often use during online interactions, it will not be able to recognise the item that the user is looking for. This generates an erroneous recommendation.

1.10.8 Long tail

If an item initially is not well-rated or not rated at all in an RS which follow a top-N recommendation, then over the time it will perish from the recommendation catalogue. A user will miss recommendations for many necessary items just because he did not rate those items or did not have any access to them. This generally leads to the long tail problem (LT). It occurs when many items remain unrated or low rated.

1.10.9 Black-box problem

The efficiency of the RS is enhanced with the increase in the transparency of recommendation. The satisfaction of the user in the recommendation is entangled with the trust that the user places on the objectives of recommendation. The black box problem occurs in RSs when the system is opaque towards the end user, causing decreased levels of confidence in the system.

1.11 Sequential Data

Sequential data are present in very different fields. There are sequences with a chronological order such as in device control where we have sequences of successive activity events, management where we have sequences of successive goods bought by customers or sequences of types of activity carried out by employees, in web usage analysis where we have sequences of visited pages , and in life course studies where we have sequences describing work careers or family life trajectories. In other domains sequences do not have a time dimension. This is for example the case of sequences of proteins or nucleotides, or of sequences of letters and words in texts [15].

1.12 Sequential recommender systems (SRSs)

The sequential recommendation is also often referred to as session-based, session-aware, or sequence-aware recommendation [16].

A Sequence Aware Recommender System is a recommender system that models the dynamics in user's interaction by extracting of sequential features from the logs of historical user activity, with the ultimate goal of adapting to users' short-term needs and providing high quality suggestions on the following item(s) the user should interact [17].

Generally, an SRS takes a sequence of user-item interactions as the input and tries to predict the subsequent user-item interactions that may happen in the near future through modelling the complex sequential dependencies embedded in the sequence of user-item interactions. More specifically, given a sequence of user-item interactions, a recommendation list consisting of top ranked candidate items are generated[18].

1.13 Difference Between Sequential Recommender Systems (SRSs) and Traditional Recommender Systems (RSs)

Different from the general sequence modelling in which the sequence structure is much simpler since a sequence is often composed of atomic elements (e.g., real values, genes), the learning task in SRSs is much more challenging because of the more complex sequence structure [18].

In traditional item recommendation, the recommender system has to infer the general user's preference over the individual items in order to suggest her items that she may like to explore. Recommendations are usually provided in the form of a list of items ordered by relevance. In contrast, SRSs treat the user-item interactions as a dynamic sequence and take the sequential dependencies into account to capture the current and recent preference of a user for more accurate recommendation [17].

1.14 Representative Tasks

Before formally defining the sequential recommendation tasks, we firstly summarize the two representative tasks in the literature (as depicted in Figure 1.4): next-item recommendation and next basket recommendation. In **next-item recommendation,** a behavior contains only one object (i.e., item), which could be a product, a song, a movie, or a location. In contrast, in **next-basket recommendation,** a behavior contains more than one object [19].

1.15 Inputs, Outputs, Computational Tasks

Inputs. The main input to sequence-aware recommendation problems is an ordered and often timestamped list of past user actions. Users can be already known by the system or anonymous ones. Each action can be associated with one of the recommendable items. Finally, each action can be of one of several pre-defined types and each action, user, and item may have a number of additional attributes. Overall, the inputs can be considered as a sort of enriched click stream data [20].

Outputs. The output of a sequence-aware recommender are ordered lists of items. In this general form, the outputs are similar to those of a traditional “item-ranking”

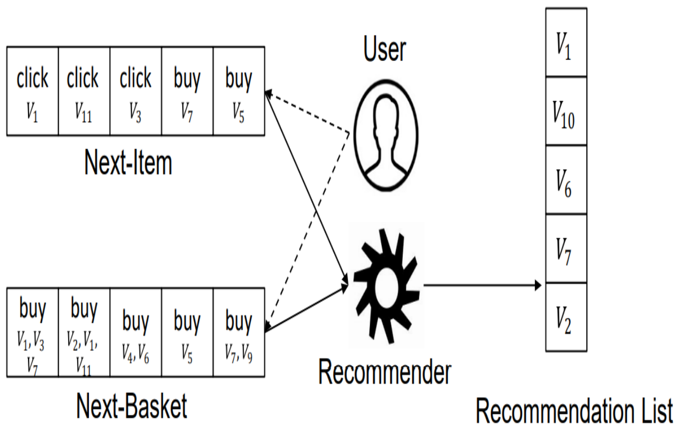

Figure 1.4: Next-item and next-basket recommendation.

recommendation setup. However, in some sequence-aware recommendation scenarios, the ordering of the objects in the recommendation list can be relevant as well [20].

Computational Tasks. Different computational tasks of sequence-aware recommenders can be identified in the literature. Most commonly, a task that is not present in traditional matrix completion setups is the identification of sequence-related patterns in the recorded user actions. These can be sequential patterns, where the order of the actions is relevant, or they can be co-occurrence patterns, where it is only important that two actions happened together [20].

1.16 Specific Computational Tasks

In order to filter and rank the items in the resulting output, different types of computations are usually done by sequential recommenders, and the particularities of these calculations again depend on the specific application scenario [17]. Figure 1.5 gives a high-level overview of the SRS problem [20]:

Beside Computational Tasks, as described in the previous section, a variety of specific Computational Tasks can be tackled with sequential recommenders. We present here an overview of what we believe are possible Specific Computational Tasks that can be accomplished with SRS [17]:

1.16.1 Context adaptation

Context-adaptation can be obtained by extracting frequent sequential patterns from the historical activity of users and then by mapping it to the most recent user actions.

Context-adaptation with sequential recommenders can be classified in the following three categories:

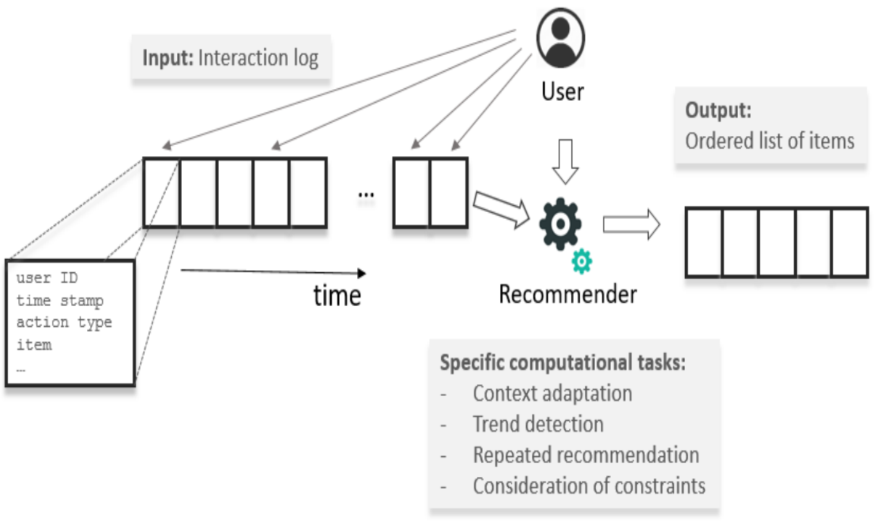

FIGURE 1.5: High-level Overview of Sequential Recommendation Problems.

Last-N interactions. This is the simplest form of context-adaptation and it is based on the sequence of the most recent N actions of the user.

Session-based recommendation. In this case recommendation depends exclusively on the current session. Session-based recommendation is common in systems having new or anonymous users – i.e. users not registered or logged into the system. For these kind of users, the only information that is available is the sequence of interactions in the current session. No historical data is available.

Session-aware recommendation. In this case we instead focus on returning users, i.e. the user is not new to the system and the recommender system can determine his historical, long-term preferences from his past activity.

1.16.2 Trend Detection

The detection of trends is another potential, but less explored, goal that can be accomplished by sequential recommenders. We can distinguish between the following types of information that can be extracted from sequential log information to be used in the recommendation process.

Community trends. Considering the popularity of items within a user community can be important for successful recommendations in practice. Sequential recommenders can aim to detect and utilize popularity patterns in the interaction logs to improve the recommendations.

Individual trends. Changes in the interest in certain items can also happen at an individual level. These interest changes can be caused when there is a “natural” interest drift, e.g., when users grow up, or when their preferences change over time.

1.16.3 Repeated Recommendation

In several application domains, recommending items that the user already knows, has consumed or purchased in the past can be meaningful.The following categories of repeated recommendation scenarios can be identified .

Identifying repeated user behavior patterns. Past interaction logs can be used by sequence-aware recommenders to identify patterns of repeated user behavior.

Repeated recommendation as reminders. In a different scenario, repeated recommendations can help to remind users of things they were interested in the past.

1.16.4 Order constraints

In certain domains, the characteristics of items may pose a sort of ordering between interactions. For example, it is reasonable to think that users that have watched a certain movie will watch its sequels later on.

1.17 Overview of Sequential Recommendation

In this section, we provide a comprehensive overview of the sequential recommendation [19].

1.17.1 Experience-based behavior sequence

same object , different behavior types. The goal is to predict next item under a target action type.

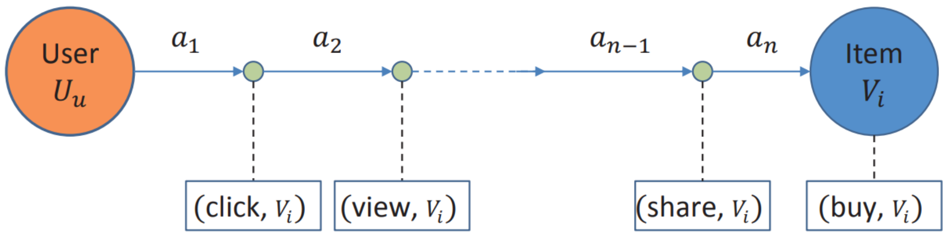

Figure 1.6: Experience-based behavior sequence.

1.17.2 Transaction-based behavior sequence

different items, same behavior type (i.e. buy). The goal is to predict next action object.

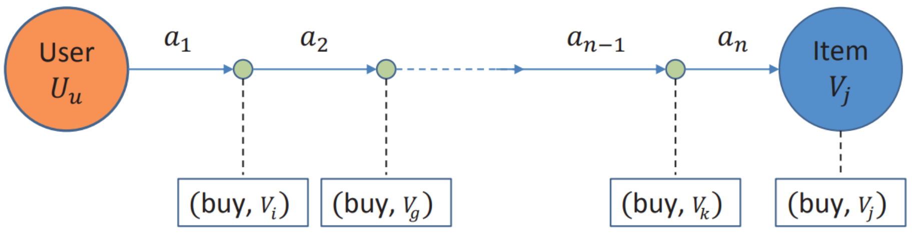

FIGURE 1.7: Transaction-based behavior sequence.

1.17.3 Interaction-based behavior sequence

multiple items, different behavior types. The goal is to predict both the action type and the action object.

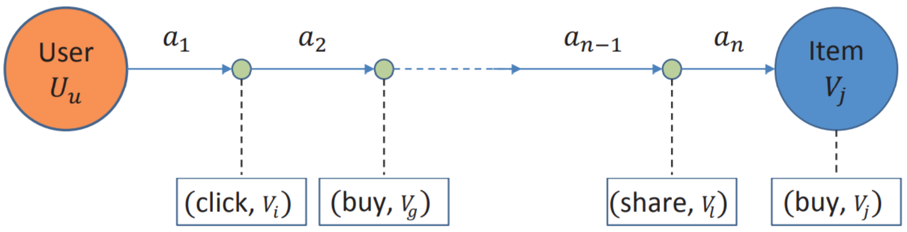

FIGURE 1.8: Interaction-based behavior sequence.

1.18 Sequential Recommender Systems approaches

In this section we present a categorization of all the approaches for SRSs [18]:

1.18.1 Traditional Sequence Models for SRSs

Traditional sequence models including sequential pattern mining and Markov chain models are intuitive solutions to SRSs by taking advantage of their natural strength in modelling sequential dependencies among the user-item interactions in a sequence.

Sequential pattern mining. Sequential pattern-based RSs first mine frequent patterns on sequence data and then utilize the mined patterns to guide the subsequent recommendations. Although simple and straightforward, sequential pattern mining usually generates a large number of redundant patterns, which increases unnecessary cost w.r.t. time and space.

Markov chain models. Markov chain-based RSs adopt Markov chain models to model the transitions over user-item interactions in a sequence, for the prediction of the next interaction. According to the specific technique used, Markov chain-based RSs are divided into basic Markov Chain-based approaches and latent Markov embedding-based approaches.

1.18.2 Latent Representation Models for SRSs

Latent representation models first learn a latent representation of each user or item, and then predict the subsequent user-item interactions by utilizing the learned representations. As a result, more implicit and complex dependencies are captured in a latent space, which greatly benefits the recommendations.

Factorization machines. Factorization machine-based SRSs usually utilize the matrix factorization or tensor factorization to factorize the observed user-item interactions into latent factors of users and items for recommendations.

Embedding. Embedding-based SRSs learn a latent representations for each user and item for the subsequent recommendations by encoding all the user-item interactions in a sequence into a latent space. Specifically, some works take the learned latent representations as the input of a network to further calculate an interaction score between users and items, or successive users' actions, while other works directly utilize them to calculate a metric like the Euclidean distance as the interaction score. This model has shown great potential in recent years due to its simplicity, efficiency and efficacy.

1.18.3 Deep Neural Network Models for SRSs

Deep neural networks have natural strength to model and capture the comprehensive relations over different entities (e.g., users, items, interactions) in a sequence, and thus they nearly dominate SRSs in the past few years.

Basic Deep Neural Networks

The most commonly used deep neural networks for SRSs are recurrent neural networks (RNN) due to their natural strength in sequence modelling, but they also have defects. Recently convolutional neural networks (CNN) and graph neural networks (GNN) have also been applied in SRSs to make up the defects of RNN.

RNN-based SRSs. Given a sequence of historical user-item interactions, an RNN-based SRS tries to predict the next possible interaction by modelling the sequential dependencies over the given interactions. Except for the basic RNN, long short-term-memory (LSTM)-based and gated recurrent unit (GRU)-based, RNN have also been developed to capture the long-term dependencies in a sequence.

CNN-based SRSs. Different from RNN, given a sequence of user-item interactions, a CNN first puts all the embeddings of these interactions into a matrix, and then treats such a matrix as an "image" in the time and latent spaces. Finally, a CNN learns sequential patterns as local features of the image using convolutional filters for the subsequent recommendations. Since a CNN does not have strong order assumptions over the interactions in a sequence, and they learn patterns between the areas in an "image" rather than over interactions, therefore, CNN-based SRSs can make up the aforementioned drawbacks of RNN-based SRSs to some degree. However, CNN-based SRSs cannot effectively capture long-term dependencies due to the limited sizes of the filters used in CNN, which limits their applications.

GNN-based SRSs. Recently, with the fast development of GNN, GNN-based SRSs have been devised to leverage GNN to model and capture the complex transitions over user-item interactions in a sequence. Typically a directed graph is first built on the sequence data by taking each interaction as a node in the graph while each sequence is mapped to a path. Then, the embeddings of users or items are learned on the graph to embed more complex relations over the whole graph. Such an approach makes full use of the advantage of GNN to capture the complex relations in structured relation datasets. GNN-based SRSs have shown a great potential to provide explainable recommendations by revealing the complex relations between the recommended items and the corresponding sequential context. Such kind of SRSs are still in their early stages.

Advanced Models

To address the limitations of SRSs built on basic neural network structures, some advanced models are usually combined together with a certain kind of basic deep neural networks (e.g., RNN, CNN) to build more powerful SRSs which are able to address particular challenges.

Attention models. Attention models are commonly employed in SRSs to emphasize those really relevant and important interactions in a sequence while downplaying those ones irrelevant to the next interaction. They are widely incorporated into shallow networks and RNN to handle interaction sequences with noise.

Memory networks. Memory networks are introduced into SRSs to capture the dependencies between any historical user-item interaction and the next one directly by incorporating an external memory matrix. Such matrix enables it possible to store and update the historical interactions in a sequence more explicitly and dynamically to improve the expressiveness of the model and reduce the interference of those irrelevant interactions.

Mixture models. A mixture model-based SRS combines different models that excel at capturing different kinds of dependencies to enhance the capability of the whole model in capturing various dependencies for better recommendations.

The categorization of SRS approaches is presented in Figure 1.9.

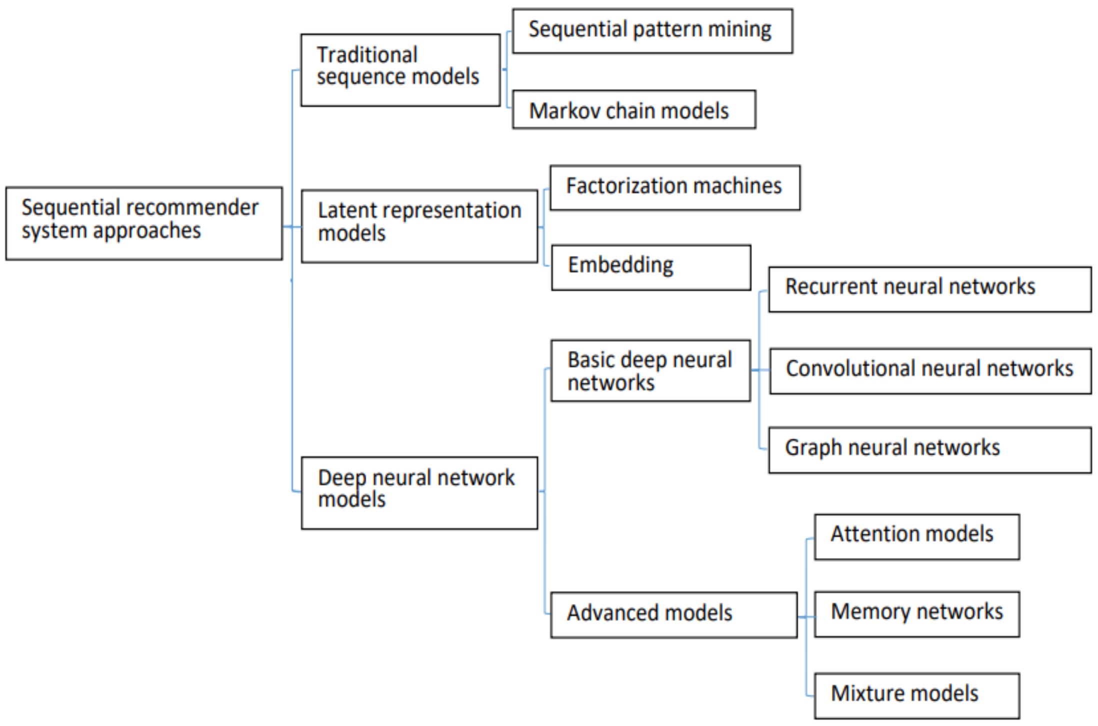
Sequential recommender system approaches
Traditional sequence models
Sequential pattern mining
Markov chain models
Latent representation models
Factorization machines
Embedding
Deep neural network models
Basic deep neural networks
Recurrent neural networks
Convolutional neural networks
Graph neural networks
Advanced models
Attention models
Memory networks
Mixture models

FIGURE 1.9: A categorization of SRS approaches.

1.19 Evaluation Protocols of Sequential Recommender Systems

In this section, we present the evaluation procedures that can be used in the evaluation of sequential recommenders [17].

1.19.1 Offline evaluation

In offline evaluation, the quality of RS is measured without inquiring real users, which is usually a slow, intrusive and expensive procedure. The quality of the results obtained by an offline evaluation procedure depends strongly on the data partitioning procedure, on the evaluation protocol and on the metrics that are employed. The whole evaluation procedure must reflect as much as possible the real application scenario.

1.19.2 Online evaluation

Online evaluation analyzes the behavior of the recommender system in a real-life scenario by directly monitoring the activity of the users, typically by means of A/B testing. This allows to analyze the impact of the RS with much more detail. However, the analysis of large amounts of users on a real system is a costly procedure, and results are hardly reproducible out of the original environment used for the evaluation. Still, it is generally agreed by the community that online evaluation provides more significant insights on the real impact of a recommendation engine than what offline evaluation does.

1.19.3 Making Reliable Choices

When choosing between algorithms, it is important that we can be confidant that the algorithm that we choose will also be a good choice for the yet unseen data the system will be faced with in the future.

1.20 Conclusion

By reviewing the RS types and filtering SRS techniques we can see that research in the topic of SRS is challenging and at the same time very interesting. The plurality of algorithms and recommendation filtering methods can be used for different types of recommendations based on the domain they are applied for. The selection of particular methods to apply in a RS is based on the desired recommendation results. Thus the development of a recommendation algorithm must be accompanied by the corresponding evaluation metrics. The next chapter will present convolutional neural network witch can be used to develop SRS.

Chapter 2

Convolutional Neural Networks

"Deep learning is inspired by the way that the human brain filters information!"

2.1 Introduction

Deep learning (DL) is playing an increasingly important role in our lives. It has already made a huge impact in areas such as cancer diagnosis, precision medicine, self-driving cars, predictive forecasting, speech recognition, etc [21]. Since the re-emergence of neural networks to the forefront of machine learning, two types of network architectures have played a pivotal role: the convolutional networks, often used for vision and higher-dimensional input data; and the recurrent networks, typically used for modeling sequential data [22]. In the early history of neural networks, convolutional models were specifically proposed as a means of handling sequence data, the idea being that one could slide a 1-D convolutional filter over the data (and stack such layers together) to predict future elements of a sequence from past ones. In this chapter, we will first describe what neural network is. Next, we will explain the motivation behind using convolution neural networks.

2.2 Artificial intelligence, Machine learning, and Deep learning

First, we need to define clearly what we're talking about when we mention AI. What are artificial intelligence, machine learning, and deep learning (see Figure 2.1) [23]?

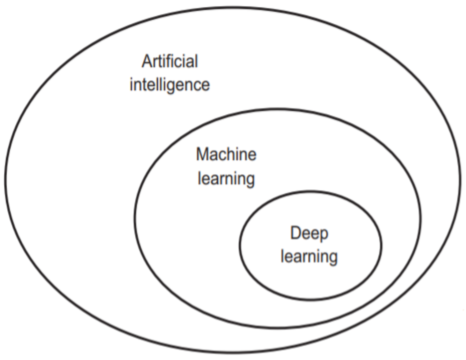

Figure 2.1: Artificial intelligence, Machine learning, and Deep learning.

2.2.1 Artificial Intelligence

Artificial intelligence isn't "new," but it's definitely experience a renaissance of sorts. And the way people use the word is also changing, much to the chagrin of traditionalists. When Turing first devised his test, the phrase artificial intelligence was largely reserved for a technology that could broadly mimic the intelligence of humans [24].

Today, the phrase artificial intelligence, or just AI, is broadly and generally used to refer to any sort of machine learning program [24].

2.2.2 Machine Learning

At its most basic level, machine learning refers to any type of computer program that can "learn" by itself without having to be explicitly programmed by a human. Today, machine learning is a widely used term that encompasses many types of programs that you'll run across in big data analytics and data mining. At the end of the day, the "brains" actually powering most predictive programs – including spam filters, product recommenders, and fraud detectors — are machine learning algorithms [24].

The Optimal Approach

"You have the data, what kind of analysis do you need [25]?"

• Regression

– predict new values based on the past, inference.

– compute the new values for a dependent variable based on the values of one or more measured attributes.

• Classification

– divide samples in classes.

– use a trained set of previously labeled data.

• Clustering

– partitioning of a data set into subsets (clusters) so that data in each subset ideally share some common characteristics.

• Classification is in a some way similar to the clustering, but requires that the analyst know ahead of time how classes are defined.

Types of Machine Learning Algorithms

Supervised learning, The user trains the program to generate an answer based on a known and labeled data set. Classification and regression algorithms, including random forests, decision trees, and support vector machines, are commonly used for supervised learning tasks [24].

Unsupervised learning, The algorithms generate answers on unknown and unlabeled data. Data scientists commonly use unsupervised techniques for discovering patterns in new data sets. Clustering algorithms, such as K-means, are often used in unsupervised machine learning [24].

Reinforcement learning, Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions. Example of Reinforcement Learning: Markov Decision Process [26].

2.2.3 Deep Learning

Deep learning is a form of machine learning that can utilize either supervised or unsupervised algorithms, or both. While it's not necessarily new, deep learning has recently seen a surge in popularity as a way to accelerate the solution of certain types of difficult computer problems, most notably in the computer vision and natural language processing (NLP) fields [24].

2.3 The Applications of Deep Learning

In recent years, there are a number of researchers have applied DL algorithms to various different fields. This section describes the fields that have been applied with Deep Learning algorithms in [27–30].

Speech Recognition (SR), Speech recognition is the very first successful application of deep learning methods at an industry scale [28]. Google has announced that Google voice search had taken a new turn by adopting Deep Neural Networks (DNN) as the core technology used to model the sounds of a language in 2012 [27].

Object Recognition and Computer Vision, Over the past two years or so, tremendous progress has been made in applying deep learning techniques to computer vision, especially in the field of object recognition. The success of deep learning in this area is now commonly accepted by the computer vision community. It is the second area in which the application of deep learning techniques is successful, following the speech recognition area as we described previously [28].

Natural Language Processing, Recently, neural network based deep learning methods have been shown to perform well on various NLP tasks such as language modeling, machine translation, part-of-speech tagging, named entity recognition, sentiment analysis, and paraphrase detection. The most attractive aspect of deep learning methods is their ability to perform these tasks without external hand-designed resources or time-intensive feature engineering [28].

Drug Discovery and Toxicology, In 2015, AtomNet has been introduced as first structure-based, deep convolutional neural network which designed to predict the bioactivity of small molecules for drug discovery applications. This paper also demonstrates how to apply the convolutional concepts of feature locality and hierarchical composition to the modeling of bioactivity and chemical interactions. AtomNet outperforms previous docking approaches on a diverse set of benchmarks by a large margin, achieving an AUC greater than 0.9 on 57.8% of the targets in the DUDE benchmark [27].

Customer Relationship Management CRM, Neural networks may be assigned to different tasks and used in different aspects of CRM systems. [30] elaborates on four examples of such applications, which exemplify practical aspects of neural networks in the field of CRM: predicting customer expenses according to behavioural data, assessing profitability of building customer relationships, assessing probability of customer leaving the company based on complaint data, determining key customers. These aspects are of great value to managers, demanding that the CRM system implemented in their company generates substantial profit and minimises the risk of a failed investment.

Recommendation Systems, Deep Learning is one of the next big things in Recommendation Systems technology. The past few years have seen the tremendous success of deep neural networks in a number of complex machine learning tasks such as computer vision, natural language processing and speech recognition. After its relatively slow uptake by the recommender systems community, deep learning for recommender systems became widely popular in 2016 [29].

2.4 Artificial Neural Networks

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks [31].

Artificial neural networks are information processing structures providing the (often unknown) connection between input and output data by artificially simulating the physiological structure and functioning of human brain structures (see Figure 2.2) [32].

2.4.1 Networks

One efficient way of solving complex problems is following the lemma “divide and conquer”. A complex system may be decomposed into simpler elements, in order to be able to understand it. Networks are one approach for achieving this. There are a large

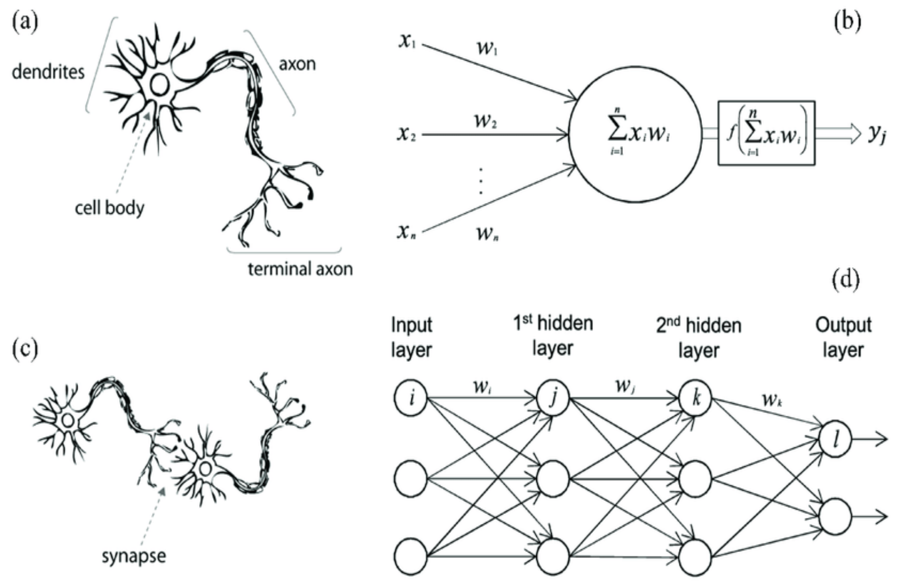

Figure 2.2: A-biological-neuron-in-comparison-to-an-artificial-neuron.

number of different types of networks, but they all are characterized by the following components [33]: a set of nodes, and connections between nodes.

The nodes can be seen as computational units. They receive inputs, and process them to obtain an output.

The connections determine the information flow between nodes. They can be unidirectional, when the information flows only in one sense, and bidirectional, when the information flows in either sense.

2.4.2 Structure of a Neural Network

Artificial neural networks are composed of elementary computational units called neurons (McCulloch & Pitts, 1943) combined according to different architectures. For example, they can be arranged in layers (multi-layer network), or they may have a connection topology [32]. Layered networks consist of:

• Input layer, made of n neurons (one for each network input);

• Hidden layer, composed of one or more hidden (or intermediate) layers consisting of m neurons;

• Output layer, consisting of p neurons (one for each network output).

The connection mode allows distinguishing between two types of architectures:

• The feedback architecture, with connections between neurons of the same or previous layer;

• The feedforward architecture, without feedback connections (signals go only to the next layer's neurons).

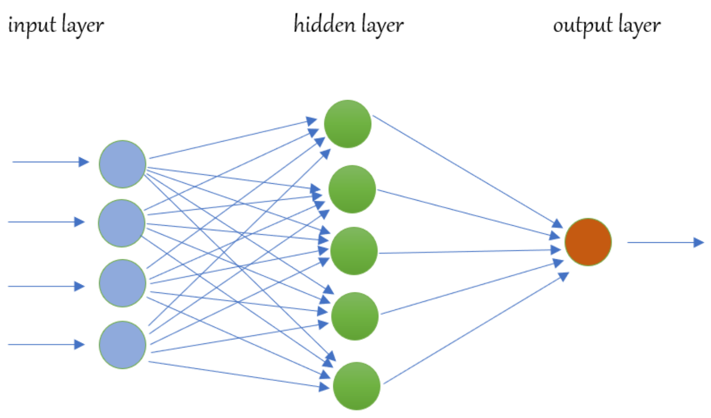

FIGURE 2.3: Artificial Neural Network Architecture.

2.4.3 Neurons

Mc Culloch and Pitts, proposed in 1943, one of the first computational models of the biological neurons. Figure 2.4 illustrates the operation of each proposed neural computational unit. As illustrated in Figure 2.4, a *neuron* N_j receives inputs from n other previous neurons $x_1, x_2, ..., x_n$. The output of each neuron $x_1, x_2, ..., x_n$ in the previous layer is multiplied by the corresponding synaptic *weight* $w_{1j}, w_{2j}, ..., w_{nj}$, also know as synaptic efficacy. The combined weighted input is transformed mathematically using a certain non-linear transfer function or an $activationfunction$ φ, generating an output o_j . In the original Mc Culloch and Pitts' neural model the activation function was a thresholding gate, giving as neural output a digital signal . This digital output neuron was the core of the first generation of neural networks [34].

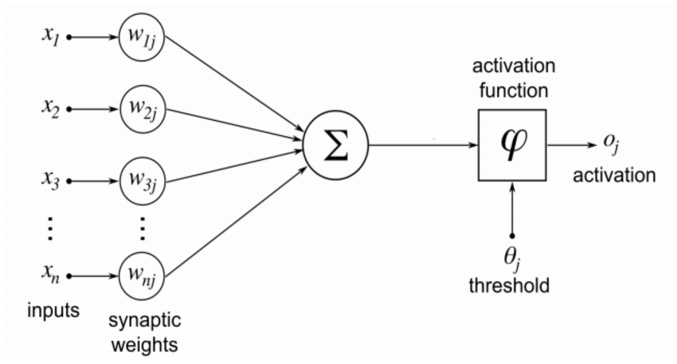

FIGURE 2.4: Diagram of an artificial neuron with n inputs with their corresponding synaptic weights. All weighted inputs are added and an activation function controls the generation of the output signal.

2.4.4 Perceptron

In 1958, Rosenblatt proposed the perceptron. The architecture of the perceptron is shown in Figure 2.5. In Figure 2.5, the computational units or neurons are represented by circles, interconnected through trainable weights representing the synaptic connections.

The original perceptron consisted of a single layer of input neurons fully interconnected in a feedforward way to a layer of output neurons. This single layer perceptron was able to solve only linearly separable problems [34].

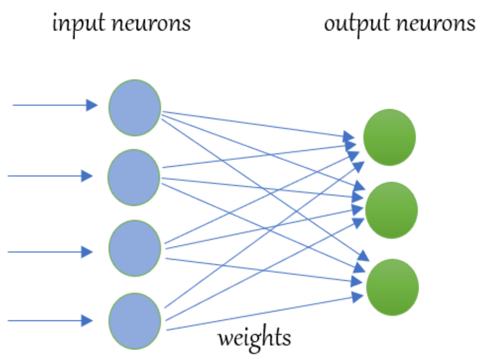

FIGURE 2.5: Architecture of a single layer perceptron.

2.4.5 Multi Layer Perceptron

In section 2.4.3,our general neuron is introduced like processing unit [35]:

$$a = \varphi\,(\sum_j w_j x_j + b)$$

where the x_j are the inputs to the unit, the w_j are the weights, b is the bias, φ is the nonlinear activation function, and a is the unit's activation.

For now, we'll concern ourselves with **feed-forward neural networks**, where the units are arranged into a graph without any cycles, so that all the computation can be done sequentially. This is in contrast with **recurrent neural networks**, where the graph can have cycles, so the processing can feed into itself [35].

The simplest kind of **feed-forward network** is a multilayer perceptron (MLP), as shown in Figure 2.6. Here, the units are arranged into a set of layers, and each layer contains some number of identical units. Every unit in one layer is connected to every unit in the next layer; we say that the network is fully connected. The first layer is the input layer, and its units take the values of the input features. The last layer is the output layer, and it has one unit for each value the network outputs (i.e. a single unit in the case of regression or binary classifiation, or K units in the case of K-class classification). All the layers in between these are known as hidden layers, because we don't know ahead of time what these units should compute, and this needs to be discovered during learning. The units in these layers are known as input units, output units, and hidden units, respectively. The number of layers is known as the depth, and the number of units in a layer is known as the width. Terminology for the depth is very inconsistent. A network with one hidden layer could be called a one-layer, two-layer, or three-layer network, depending if you count the input and output layers. As you might guess, “deep learning” refers to training neural nets with many layers [35].

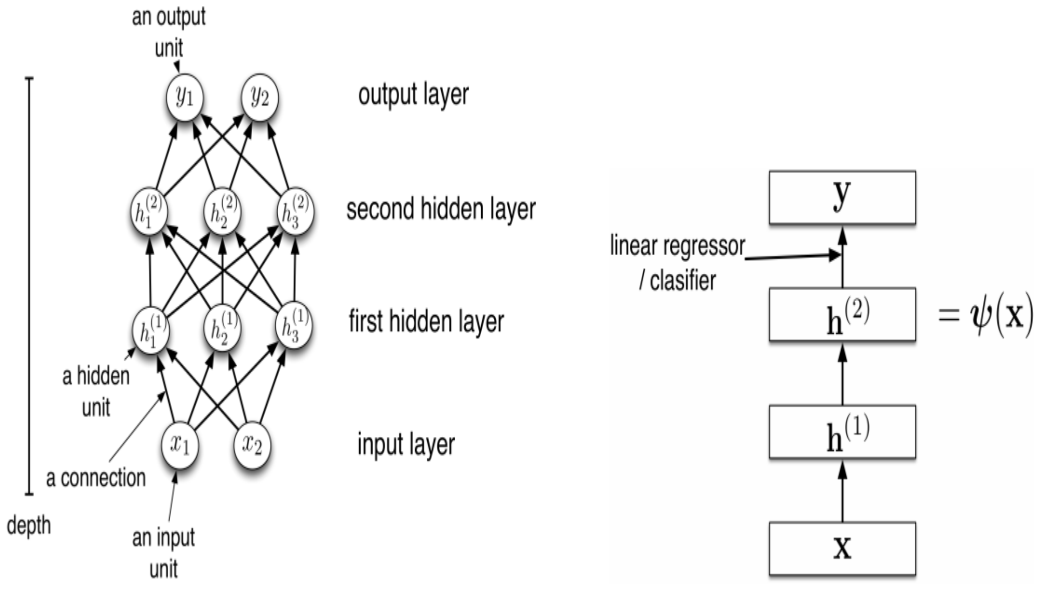

Figure 2.6: A multilayer perceptron with two hidden layers. **Left:** with the units written out explicitly. **Right:** representing layers as boxes.

2.4.6 Training Algorithms

The learning algorithm constitutes the main part of Deep Learning. The number of layers differentiates the deep neural network from shallow ones. The higher the number of layers, the deeper it becomes. Each layer can be specialized to detect a specific aspect or feature [36].

The goal of the learning algorithm is to find the optimal values for the weight vectors to solve a class of problem in a domain. Some of the well-known training algorithms are [36]:

1. Gradient Descent
2. Stochastic Gradient Descent
3. Momentum
4. Levenberg–Marquardt algorithm
5. Backpropagation through time

Gradient Descent

Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems. The general idea of Gradient Descent is to tweak parameters iteratively in order to minimize a cost function [37].

Suppose you are lost in the mountains in a dense fog; you can only feel the slope of the ground below your feet. A good strategy to get to the bottom of the valley quickly is to go downhill in the direction of the steepest slope. This is exactly what Gradient Descent does: it measures the local gradient of the error function with regards to the parameter vector θ, and it goes in the direction of descending gradient. Once the gradient is zero, you have reached a minimum [37]!

Concretely, it start by filling θ with random values (this is called random initialization), and then improve it gradually, taking one baby step at a time, each step attempting to decrease the cost function (e.g., the MSE), until the algorithm converges to a minimum (see Figure 2.7) [37].

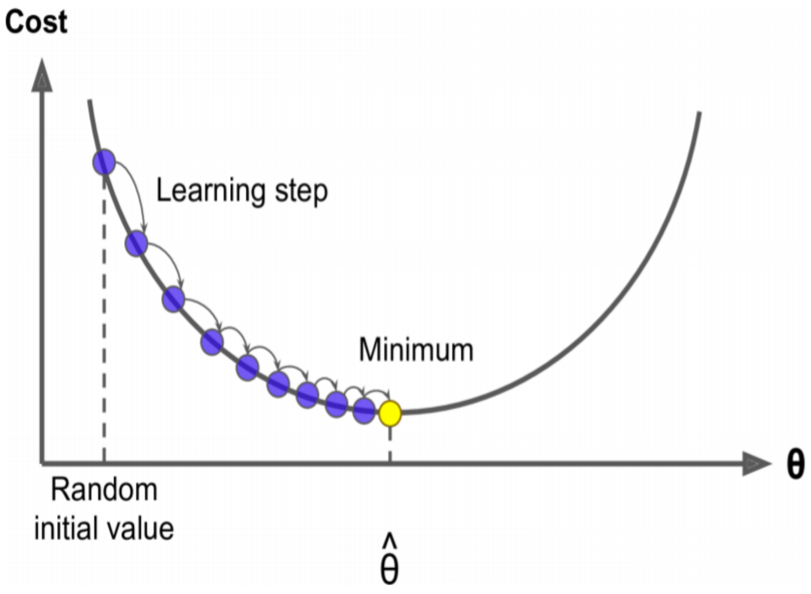

FIGURE 2.7: Gradient Descent.

Cost function is one half the square of the difference between the desired output minus the current output as shown below [36].

$$C = \frac{1}{2}\left(y_{expected} - y_{actual}\right)^2$$

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is the most common variation and implementation of gradient descent. In gradient descent, the process go through all the samples in the training dataset before applying the updates to the weights. While in SGD, updates are applied after running through a minibatch of n number of samples. Since the weights are updating more frequently in SGD than in GD, it can converge towards global minimum much faster [36].

Momentum

In the standard SGD, learning rate is used as a fixed multiplier of the gradient to compute step size or update to the weight. This can cause the update to overshoot a potential minima, if the gradient is too steep, or delay the convergence if the gradient is noisy. Using the concept of momentum in physics, the momentum algorithm presents a velocity V variable that configured as an exponentially decreasing average of the gradient. This helps prevent costly descent in the wrong direction [36].

Levenberg-Marquardt algorithm

Levenberg-Marquadt algorithm (LMA) is primarily used in solving non-linear least squares problems such as curve fitting. In least squares problems, we try to fit a given data points with a function with the least amount of sum of the squares of the errors between the actual data points and points in the function. LMA uses a combination of gradient descent and Gauss-Newton method. Gradient descent is employed to reduce the sum of the squared errors by updating the parameters of the function in the direction of the steepest-descent, while the Gauss-Newton method minimizes the error by assuming the function to be locally quadratic and finds the minimum of the quadratic [36].

Backpropagation through time

Backpropagation through time (BPTT) is the standard method to train the recurrent neural network. the unrolling of RNN in time makes it appears like a feedforward network. But unlike the feedforward network, the unrolled RNN has the same exact set of weight values for each layer and represents the training process in time domain. The backward pass through this time domain network calculates the gradients with respect to specific weights at each layer. It then averages the updates for the same weight at different time increments (or layers) and changes them to ensure the value of weights at each layer continues to stay uniform [36].

There exist also more sophisticated algorithms, called adaptive algorithms. One of the most famous is the RMSProp algorithm, or Adam (for Adaptive Moments) algorithm.

2.5 Recurrent Neural Networks

Another category of neural networks is the recurrent neural networks. In this type, the hidden layer saves its output to be used for future prediction. The output becomes part of its new input. The key feature of the RNN is that it allows us to process sequential data and exploit the dependency among data. RNNs are widely used in language related tasks such as the language modeling, text generating, machine translation, speech recognition and image caption generation. Commonly used RNN structures include bi-directional RNNs, LSTM networks and deep RNNs. Given a sequence of input data $x = x_1, x_2, ..., x_T$, a standard RNN compute the hidden vector sequence $h = h_1, h_2, ..., h_T$ and the output sequence $y = y_1, y_2, ..., y_T$ for every time step $t = 1, 2, ..., T$ as:

$$h_t = f(W_{ih}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = W_{ho}h_t + b_o$$

where W_{ih}, W_{hh}, W_{ho} and b_h, b_o are the weight matrices and the bias vectors. f is the activation function of the hidden layer, A recurrent neural network can be seen as multiple copies of the same neural network, each passing a value to its successor [38].

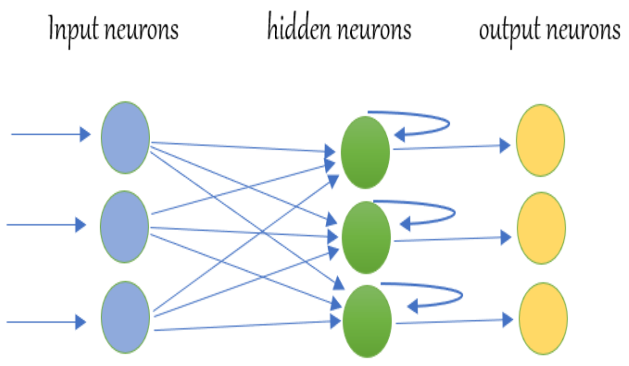

FIGURE 2.8: Recurrent Neural Network.

2.6 Convolutional Neural Networks

Convolutional networks, also known as convolutional neural networks, convnets or CNNs for short, are a specialized kind of neural networks for processing data that has a known, grid-like topology. Examples include time-series data, which can be thought of as a 1D grid taking samples at regular time intervals, and image data, which can be thought of as a 2D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution [39].

2.6.1 kernels

A kernel can be described as a grid of discrete values or numbers, where each value is known as the weight of this kernel. During the starting of training process of an CNN model, all the weights of a kernel are assigned with random numbers (different approaches are also available there for initializing the weights). Then, with each training epoch, the weights are tuned and the kernel learned to extract meaningful features [40].

2.6.2 Convolution

Before we go any deeper, let us first understand the convolution operation, If we take a gray scale image of 4x4 dimension, and an 2x2 kernel with randomly initialized weights (see Figure 2.9) [40].

Now, convolution operation, take the 2x2 kernel and slide it over all the complete 4x4 image horizontally as well as vertically and along the way it take the dot product between kernel and input image by multiplying the corresponding values of them and sum up all values to generate one scaler value in the output feature map. This process continues until the kernel can no longer slide further [40].

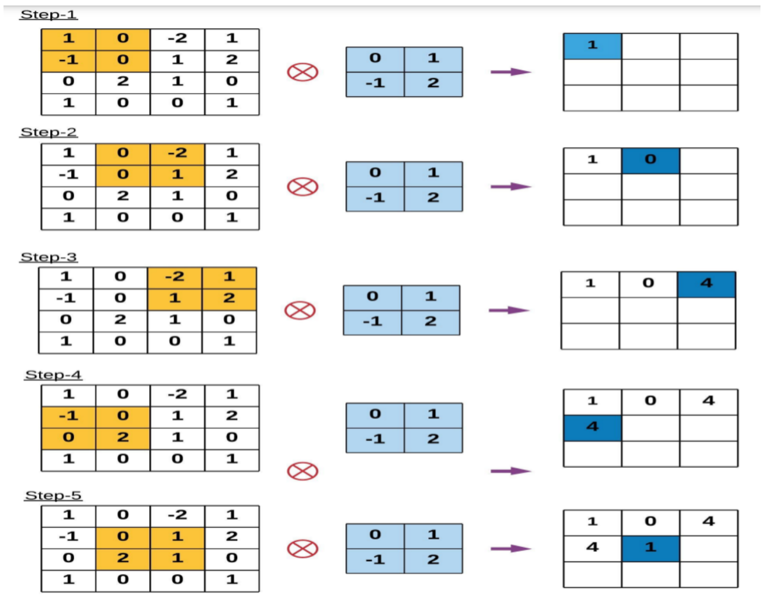

FIGURE 2.9: Illustrating the first 5 steps of convolution operation.

2.6.3 Convolutional Neural Networks Architecture

Over the last years, Convolutional Neural Networks (CNN) have yielded state-of-the-art results for a wide variety of tasks in the field of computer vision, such as object classification, traffic sign recognition and image caption generation. The use of CNN was not limited to the field of computer vision, and these models were adapted successfully to a variety of audio processing and natural language processing tasks such as speech recognition, and sentence classification. When applying a convolutional layer on some data, this layer is extracting features from local patches of the data and often is followed by a pooling mechanism to pool values of features over neighboring patches. The extracted features can be the input of the next layer in a neural network, possibly another convolutional layer or a classifier. Models using convolutional layers for extracting features from raw data can outperform models using hand-crafted features and achieve state-of-the-art-results [41].

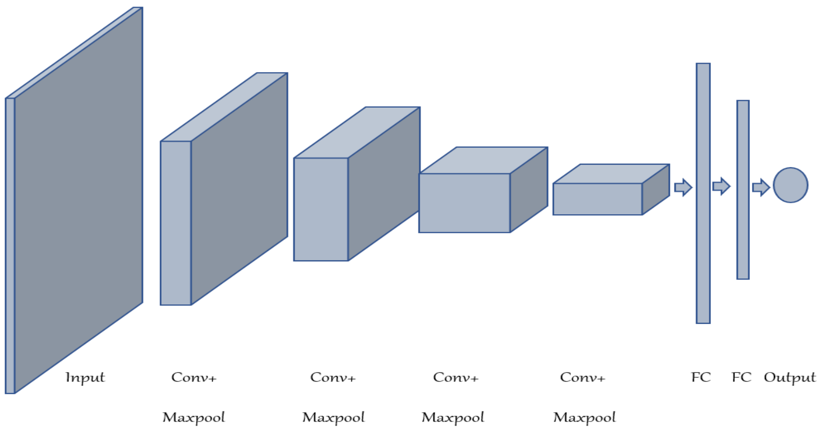

FIGURE 2.10: A basic architecture of a Convolutional Neural Network.

2.6.4 Convolutional Neural Networks Types

1D Convolutional Neural Network

In Conv 1D, the kernel slides in one dimension.

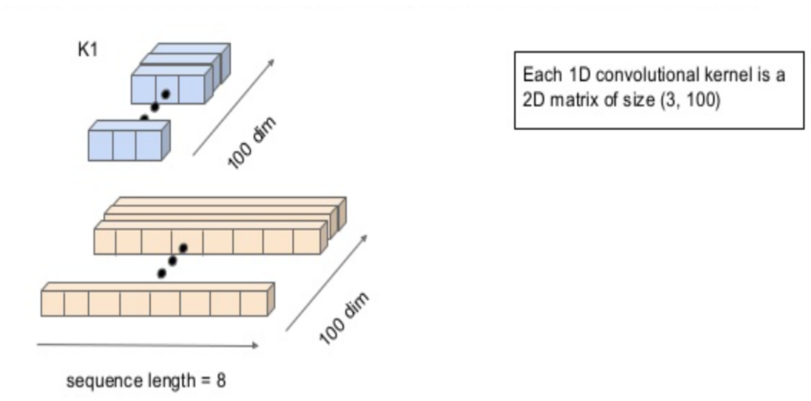
K1
100 dim
100 dim
sequence length = 8
Each 1D convolutional kernel is a 2D matrix of size (3, 100)

FIGURE 2.11: 1D Convolutional Neural Network.

2D Convolutional Neural Network

This is the first standard convolutional neural network introduced in the Lenet-5 architecture. Conv 2D is usually used for image data. It is called a 2D CNN because the kernel slides along the data in 2 dimensions, as shown in the figure below.

3D Convolutional Neural Network

In Conv3D, the kernel slides in 3 dimensions, as shown below.

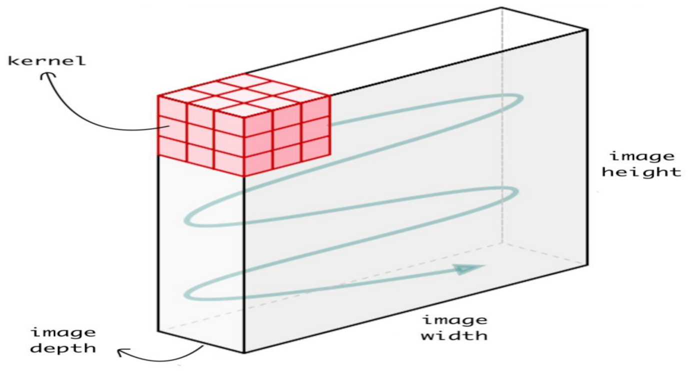

FIGURE 2.12: 2D Convolutional Neural Network.

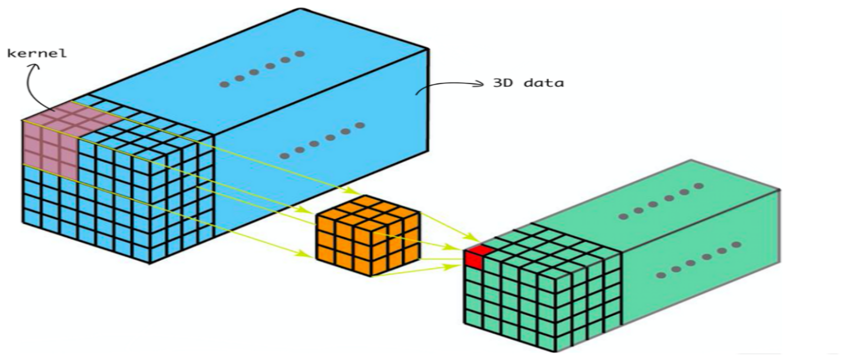

FIGURE 2.13: 3D Convolutional Neural Network.

The three types are shown in (Figure 2.14) .

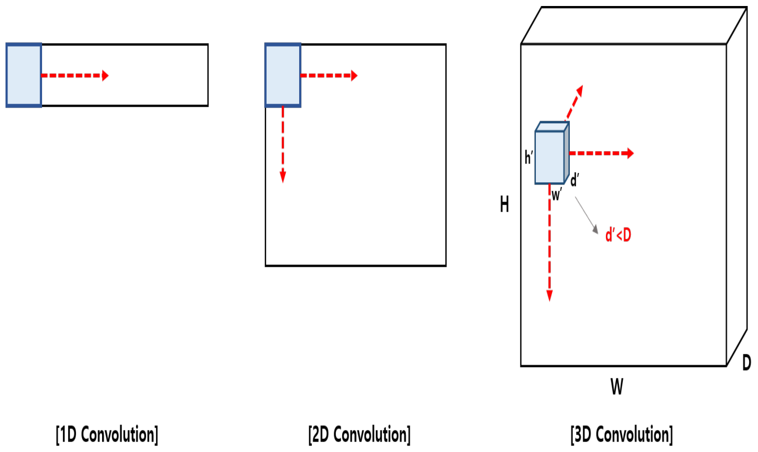

FIGURE 2.14: 1D, 2D and 3D Convolutional Neural Network.

2.7 Features Engineering

Feature engineering is the process of using your own knowledge about the data and about the machine-learning algorithm at hand (in this case, a neural network) to make the algorithm work better by applying hardcoded (nonlearned) transformations to the data before it goes into the model. In many cases, it isn't reasonable to expect a machinelearning model to be able to learn from completely arbitrary data. The data needs to be presented to the model in a way that will make the model's job easier [3].

Fortunately, modern deep learning removes the need for most feature engineering, because neural networks are capable of automatically extracting useful features from raw data. Does this mean you don't have to worry about feature engineering as long as you're using deep neural networks? No, for two reasons [3]:

• Good features still allow you to solve problems more elegantly while using fewer resources. For instance, it would be ridiculous to solve the problem of reading a clock face using a convolutional neural network.

• Good features let you solve a problem with far less data. The ability of deep-learning models to learn features on their own relies on having lots of training data available; if you have only a few samples, then the information value in their features becomes critical.

2.8 Layers

The fundamental data structure in neural networks is the layer, to which we will introduce in this section. A layer is a data-processing module that takes as input one or more tensors and that outputs one or more tensors. A CNN layer typically includes 3 operations: convolution, activation and pooling .Some layers are stateless, but more frequently layers have a state: the layer's weights, one or several tensors learned with stochastic gradient descent, which together contain the network's knowledge [42]. Different layers are appropriate for different tensor formats and different types of data processing. For instance, simple vector data, stored in 2D tensors of shape (samples, features), is often processed by densely connected layers, also called fully connected or dense layers (the Dense class in Keras). Sequence data, stored in 3D tensors of shape (samples, timesteps, features), is typically processed by recurrent layers such as an LSTM layer. Image data, stored in 4D tensors, is usually processed by 2D convolution layers (Conv2D) [42].

2.8.1 Convolution Layer

The convolution operation is one of the fundamental building blocks of a convolutional neural network. The convolutional layer's parameters consist of a set of learnable filters (kernels). Every filter is small spatially (along width and height), but extends through the full depth of the input volume. Typical filter sizes might have size 3x3, 5x5, 7x7. The third dimension of the filter corresponds to the number of channels in the input. The grayscale image depth is 1 and the color image has 3 (RGB) color channels [43].

During the forward propagation, each filter performs convolution on the input volume across the width and height and compute the dot products between the entries of the filter and the input at any position, this operation is followed by a nonlinear activation function (sigmoid, tanh, ReLU etc.), the resulting outputs are called feature maps. The feature map (also known as an activation map), gives the responses of the filter at every spatial position. An example of convolution layer followed by nonlinear activation is shown in Figure 2.15 We stack these activation maps along the depth dimension and produce the output volume. The output volume depends on three hyperparameters: depth, stride and padding [43].

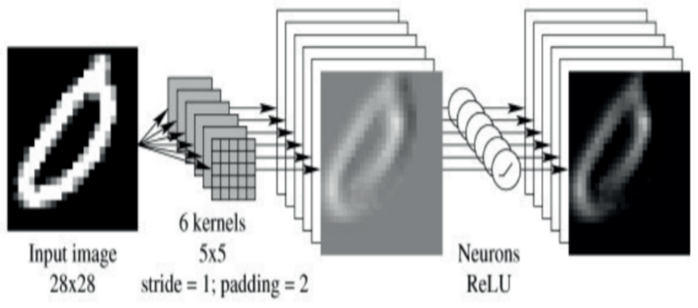

Figure 2.15: Convolution Layer.

• The depth of the output volume represents the number of filters that are used in the convolution operation. Each filter is learning something different in the input, edges, blobs, colors.

• The stride is the number of steps that we slide the filter in the input. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 then the filters jump 2 pixels at a time as we slide them around. Th is will produce smaller output volumes spatially.

• Padding allows controlling the output size. Applying convolution to an input, reduce the output size that leads to losing information. To avoid that, we pad the input volume with zeros around the border. Two common choices are valid convolution and the same convolution. The valid convolution means no padding, the same convolution means that the output size remains the same as the input size.

The output size is calculated in the following way:

$$(n + 2p - f)/s + 1$$

Where n is the number of filters, p is the amount of padding, f is the filter size and s is the stride.

2.8.2 Pooling Layer

CNNs often use pooling layer operation after convolution layers, its function is to reduce the dimension, also referred as subsampling or downsampling. Hyperparameters of pooling layer represent the filter size and strides. Most commonly used pooling layer is with filter size 2 and with stride 2. Two common types of pooling layers are max pooling and average pooling, where the maximum and average value is taken, respectively. And we can use average pooling instead of max pooling, where each local input patch is transformed by taking the average value of each channel over the patch, rather than the max. But max pooling tends to work better than these alternative solutions [42]. Max pooling is used often than average pooling. Pooling layer does not have parameters to learn. The intuition of what max pooling is doing is that the large number means that there may be detected a feature. An example of convolution layer followed by pooling layer is shown in Figure 2.16 [43].

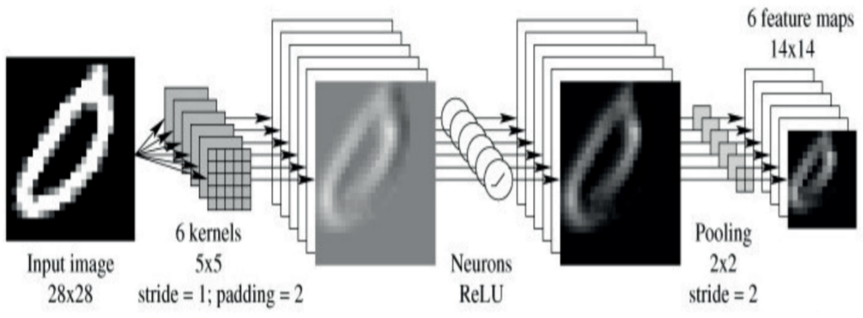

FIGURE 2.16: Convolution Layer followed by Pooling Layer.

2.8.3 Fully Connected Layer

After several convolution and pooling layers, the CNN generally ends with several fully connected layers. The tensor that we have at the output of these layers is transformed into a vector and then we add several neural network layers. The fully connected layers typically are the last few layers of the architecture as shown in the Figure 2.17, the dropout regularization technique can be applied in the fully connected layers to prevent overfitting. The final fully connected layer in the architecture contains the same amount of output neurons as the number of classes to be recognized [43].

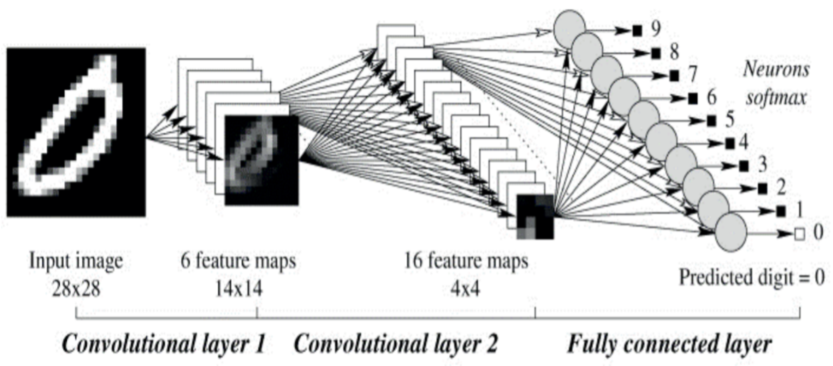

FIGURE 2.17: Two Convolutional Layers followed by a Fully Connected Layer.

2.8.4 Loss Functions

The last layer is a loss layer. Loss functions quantify the agreement between the predicted output (or label) and the ground truth output. We use loss functions to determine the penalty for an incorrect classification or regression of an input vector [44]. Let us suppose t is the corresponding target (ground-truth) value for the input x^1 , then a cost or loss function can be used to measure the discrepancy between the CNN prediction x^L and the target t. For example, a simple loss function could be:

$$z = \frac{1}{2}||t - x^L||^2$$

Although more complex loss functions are usually used. This squared l_2 loss can be used in a regression problem. In a classification problem, the cross entropy loss is often used [45].

2.9 Activation Functions

Activation functions decide whether a neuron should be activated or not by calculating the weighted sum and further adding bias with it. They are differentiable operators to transform input signals to outputs, while most of them add non-linearity. Because activation functions are fundamental to deep learning [46].

Activation functions are a choice that you must make for each layer. Generally, you can follow this guideline:

• Hidden Layers - RELU

• Output Layer - Softmax for classification, linear for regression.

• ReLU function is the most widely used function and performs better than other activation functions in most of the cases.

• ReLU function has to be used only in the hidden layers and not in the outer layer.

Without activation functions, the outputs can be anything on the ranging $[-Inf, +Inf]$, so the neurons really don't know the bounds of the value [46]. Some of the common activation functions are listed here (see Figure 2.18)[47]:

Binary Step Function

Binary Step Function is the simplest activation function that exists and it can be implemented with simple if-else statements in Python. While creating a binary classifier binary activation function are generally used. But, binary step function cannot be used in case of multiclass classification in target carriable. Also, the gradient of the binary step function is zero which may cause a hinderance in back propagation step i.e if we calculate the derivative of f(x) with respect to x, it is equal to zero. Mathematically binary step function can be defined as:

$$f(x) = 1 \ , \ x >= 0$$

$$f(x) = 0 \ , \ x < 0$$

Linear

The linear activation function is directly proportional to the input. The main drawback of the binary step function was that it had zero gradient because there is no component of x in binary step function. In order to remove that, linear function can be used. It can be defined as:

$$F(x) = ax$$

The value of variable a can be any constant value chosen by the user.

Sigmoid

It is the most widely used activation function as it is a non-linear function. Sigmoid function transforms the values in the range 0 to 1. It can be defined as:

$$f(x) = 1/e^{-x}$$

Sigmoid function is continuously differentiable and a smooth S-shaped function. The derivative of the function is:

$$f'(x) = 1 - sigmoid(x)$$

Also, sigmoid function is not symmetric about zero which means that the signs of all output values of neurons will be same. This issue can be improved by scaling the sigmoid function.

Tanh

It is Hyperbolic Tangent function. Tanh function is similar to the sigmoid function but it is symmetric to around the origin. This results in different signs of outputs from previous layers which will be fed as input to the next layer. It can be defined as:

$$f(x) = 2sigmoid(2x) - 1$$

Tanh function is continuous and differentiable, the values lies in the range -1 to 1. As compared to the sigmoid function the gradient of tanh function is more steep. Tanh is preferred over sigmoid function as it has gradients which are not restricted to vary in a certain direction and also, it is zero centered.

ReLU

ReLU stands for rectified liner unit and is a non-linear activation function which is widely used in neural network. The upper hand of using ReLU function is that all the neurons are not activated at the same time. This implies that a neuron will be deactivated only when the output of linear transformation is zero. It can be defuned mathematically as:

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

Leaky ReLU

Leaky ReLU is an improvised version of ReLU function where for negative values of x, instead of defining the ReLU functions' value as zero, it is defined as extremely small linear component of x. It can be expressed mathematically as:

$$f(x) = 0.01x\ ,\ x < 0$$

$$f(x) = x \ , \ x >= 0$$

Parametrized ReLU

It is also a variant of Rectified Linear Unit with better performance and a slight variation. It resolves the problem of gradient of ReLU becoming zero for negative values of x by introducing a new parameter of the negative part of the function i.e Slope. It is expressed as:

$$f(x) = x \ , \ x >= 0$$

$$f(x) = ax \ , \ x < 0$$

The value of a, when set to 0.01, it behaves as leaky ReLU function but here a is also a trainable parameter. For faster and optimum convergence, the network learns the value of a.

Exponential Linear Unit

Exponential Linear Unit or ELU is also a variant of Rectified Linear Unit. ELU introduces a parameter slope for the negative values of x. It uses a log curve for defining the negative values.

$$f(x) = x \ , \ x >= 0$$

$$f(x) = a(e^x - 1) \ , \ x < 0$$

Swish

Swish function is a relatively new activation function which was discovered by researchers at GOOGLE. The distinguishing feature of Swish function is that it is nit Monotonic, which means that the value of function may decrease even though the values of inputs are increasing. In some cases, Swish outperforms even the ReLU function. It is expressed mathematically as:

$$f(x) = x * sigmoid(x)$$

$$f(x) = x/(1 - e^{-x})$$

SoftMax

Softmax function is a combination of multiple sigmoid functions. As we know that a sigmoid function returns values in the range 0 to 1, these can be treated as probabilities of a particular class' data points. Softmax function unlike sigmoid functions which are used for binary classification, can be used for multiclass classification problems. The function, for every data point of all the individual classes, returns the probability. It can be expressed as:

$$\sigma(Z)_j = \frac{e^{zj}}{\sum_{k=1}^{k} e^{zk}} \qquad for\ j = 1, ..., k$$

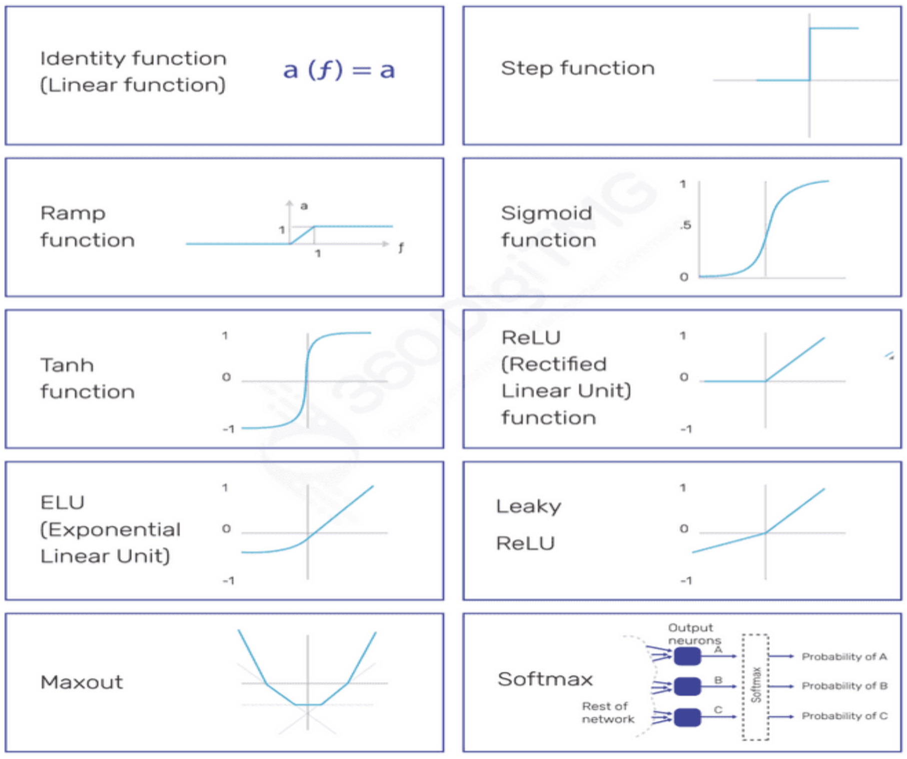

FIGURE 2.18: Common activation functions in ANN.

2.10 Convolutional Neural Networks for Sequence Data

Such 1D convnets can be competitive with RNNs on certain sequence-processing problems, usually at a considerably cheaper computational cost. Recently, 1D convnets, typically used with dilated kernels, have been used with great success for audio generation and machine translation. In addition to these specific successes, it has long been known that small 1D convnets can offer a fast alternative to RNNs for simple tasks such as text classification and timeseries forecasting [42].

The convolution layers introduced previously were 2D convolutions, extracting 2D patches from image tensors and applying an identical transformation to every patch. In the same way, we can use 1D convolutions, extracting local 1D patches (sub-sequences) from sequences (see Figure 2.19) [42].

Such 1D convolution layers can recognize local patterns in a sequence. Because the same input transformation is performed on every patch, a pattern learned at a certain position in a sentence can later be recognized at a different position, making 1D convnets translation invariant (for temporal translations). For instance, a 1D convnet processing sequences of characters using convolution windows of size 5 should be able to learn words or word fragments of length 5 or less, and it should be able to recognize these words in any context in an input sequence. A character-level 1D convnet is thus able to learn about word morphology [42].

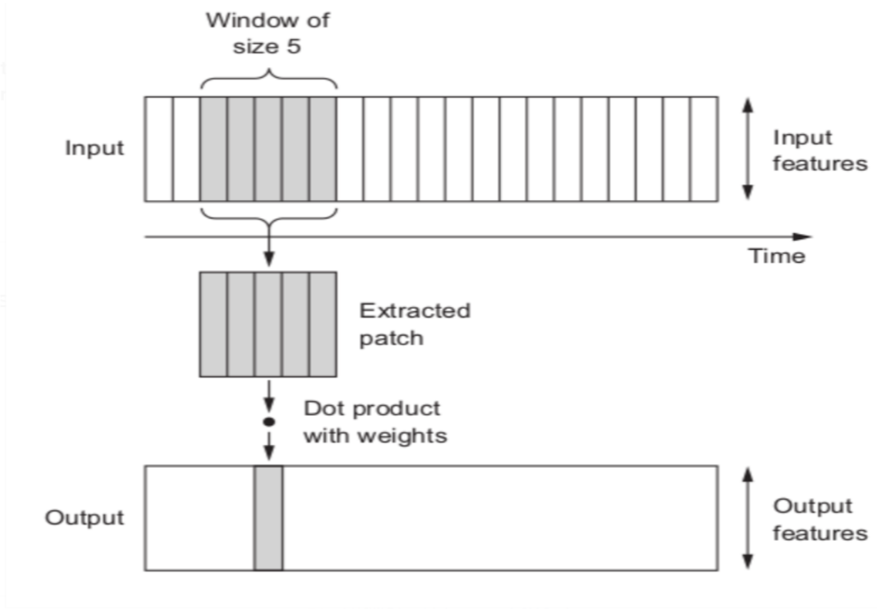

FIGURE 2.19: How 1D convolution works: each output timestep is obtained from a temporal patch in the input sequence.

2.10.1 1D pooling for sequence data

Such as 2D average pooling and max pooling, used in convnets to spatially downsample image tensors. The 2D pooling operation has a 1D equivalent: extracting 1D patches (subsequences) from an input and outputting the maximum value (max pooling) or average value (average pooling). Just as with 2D convnets, this is used for reducing the length of 1D inputs (subsampling) [42].

2.11 Why Deep Neural Networks for Recommendation?

Before diving into the details of our work, it is beneficial to understand the reasons of applying deep learning techniques to recommender systems. It is evident that numerous deep recommender systems have been proposed in a short span of several years. The field is indeed bustling with innovation. At this point, it would be easy to question the need for so many different architectures and/or possibly even the utility of neural networks for the problem domain. Along the same tangent, it would be apt to provide a clear rationale of why each proposed architecture and to which scenario it would be most beneficial for. All in all, this question is highly relevant to the issue of task, domains and recommender scenarios. One of the most attractive properties of neural architectures is that they are (1) end-to-end differentiable and (2) provide suitable inductive biases catered to the input data type. As such, if there is an inherent structure that the model can exploit, then deep neural networks ought to be useful. For instance, CNNs and RNNs have long exploited the instrinsic structure in vision (and/or human language). Similarly, the sequential structure of sequences, sessions or click-logs are highly suitable for the inductive biases provided by recurrent/convolutional models [3].

When dealing with textual data (reviews, tweets etc.), image data (social posts, product images), CNNs/RNNs become indispensable neural building blocks. Here, the traditional alternative (designing modality-specific features etc.) becomes significantly less attractive and consequently, the recommender system cannot take advantage of joint (end-to-end) representation learning [3].

Pertaining to the interaction-only setting (i.e., matrix completion or collaborative ranking problem), the key idea here is that deep neural networks are justified when there is a huge amount of complexity or when there is a large number of training instances [3].

2.12 Conclusion

In this chapter, the architecture of Convolutional Neural Networks (CNN) in conjunction with it's few applications, in brief, have been discussed. Also, the evolution of the various ANN architectures has been presented clearly along with their components. The CNN is better than other alternative deep learning networks in applications such as computer vision and natural language processing as it can mitigate the error rate significantly and hence improve network performances. By analyzing this chapter, one can gain a better understanding of why CNN is employed in numerous applications and facilitates in several machine learning fields. and finally discussed how convolutional neural network provides a unified and flexible network structure for capturing both general preferences and sequential patterns in a sequential recommender system. The next chapter will present the CNN model that will be used to build an SRS.

Chapter 3

Sequential Recommendation Using Convolutional Neural Networks

"sequence models have proved useful in recommender systems"

3.1 Introduction

Recent works have applied sequence models for dynamic recommender systems as well. These techniques usually involve using a specific sequence model, such as RNN or Convolutional Neural Networks (CNN), to encode users' past interactions into a latent feature space, which is then used for future predictions [48].

Sequence models, CNNs and RNNs are two important architectures for sequence modeling. The sequential nature of RNN has made it the default choice for sequence modeling, such 1D CNNs can be competitive with RNNs on certain sequence-processing problems, usually at a considerably cheaper computational cost. Recently, 1D CNNs, typically used with dilated kernels, have been used with great success for audio generation and machine translation. In addition to these specific successes, it has long been known that small 1D CNNs can offer a fast alternative to RNNs for simple tasks such as text classification and timeseries forecasting [48]. In this chapter we will introduce our model to show how convolutional neural networks can be used to develop a sequential recommender system.

3.2 Time Series Forecasting

Time series forecasting is difficult. Unlike the simpler problems of classification and regression, time series problems add the complexity of order or temporal dependence between observations. This can be difficult as specialized handling of the data is required when fitting and evaluating models. This temporal structure can also aid in modeling, providing additional structure like trends and seasonality that can be leveraged to improve model skill. Traditionally, time series forecasting has been dominated by linear methods like ARIMA because they are well understood and effective on many problems [49]. But these classical methods also suffer from some limitations, such as [49]:

• **Focus on complete data:** missing or corrupt data is generally unsupported.

• **Focus on linear relationships:** assuming a linear relationship excludes more complex joint distributions.

• **Focus on fixed temporal dependence:** the relationship between observations at different times, and in turn the number of lag observations provided as input, must be diagnosed and specified.

• **Focus on univariate data:** many real-world problems have multiple input variable

• **Focus on one-step forecasts:** many real-world problems require forecasts with a long time horizon.

3.3 Convolutional Neural Networks for Time Series

The ability of CNNs to learn and automatically extract features from raw input data can be applied to time series forecasting problems. A sequence of observations can be treated like a one-dimensional image that a CNN model can read and distill into the most salient elements [49].

This capability of CNNs has been demonstrated to great effect on time series classification tasks such as automatically detecting human activities based on raw accelerator sensor data from fitness devices and smartphones [49].

CNNs get the benefits of Multilayer Perceptrons for time series forecasting, namely support for multivariate input, multivariate output and learning arbitrary but complex functional relationships, but do not require that the model learn directly from lag observations. Instead, the model can learn a representation from a large input sequence that is most relevant for the prediction problem [49].

3.4 Convolutional Neural Networks for Sequential Recommendation

We generate item recommendations by learning item feature vector embeddings. Our work is similar to approaches like Word2Vec or Glove used to generate a good vector representation of words in a natural language corpus. We treat the items that a user interacted with as words and the string of items interacted with in a sequence as sentences. This ‘corpus’ of items is then used as an input to a Convolutional Neural Network. The Neural Network then learn item vector representations and capture the relationship between items and capture the patterns from sequences. The similarity between items is captured in the representation of the item vectors. We then use these item-item similarities to generate recommendations for users based on their browsing history.

3.4.1 What Are Word Embeddings?

A word embedding is a learned representation for text where words that have the same meaning have a similar representation. It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.

Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.

Key to the approach is the idea of using a dense distributed representation for each word. Each word is represented by a real-valued vector, often tens or hundreds of dimensions. This is contrasted to the thousands or millions of dimensions required for sparse word representations, such as a one hot encoding.

3.4.2 Items Embedding (Word2Vec)

We used the Word2Vec techniques implemented in the gensim toolbox. This implementation accepts a list of sentences which are themselves are a list of words. These words are used to create the internal dictionary which holds the words and their frequencies. Afterwards the model is trained using the input data and the dictionary. The output of the technique is continuous vector representation of words, which can be used as features by different applications. During the training various parameters can be tuned which affects the performance, in terms of time and quality [50].

There are several similarities between the Word2Vec techniques and the recommendation process: First, the input data used in Word2Vec techniques is actually similar to what is used in the recommendation process. In the recommendation process a list of items that the user preferred/rated in the past are used and these lists can be divided into individual items. In other words, the sentences used in Word2Vec can be mapped into past preferences of users in recommendation process and the words in Word2Vec to individual items used in recommendation process. Second, the purpose of the Word2Vec techniques and the recommendation process are similar. Word2Vec model aims to predict the words based on the observed words, which can be mapped to predicting the items to be recommended based on already preferred/used items [50].

In Figure 3.1 An illustrative example of Word2Vec embedding is shown.

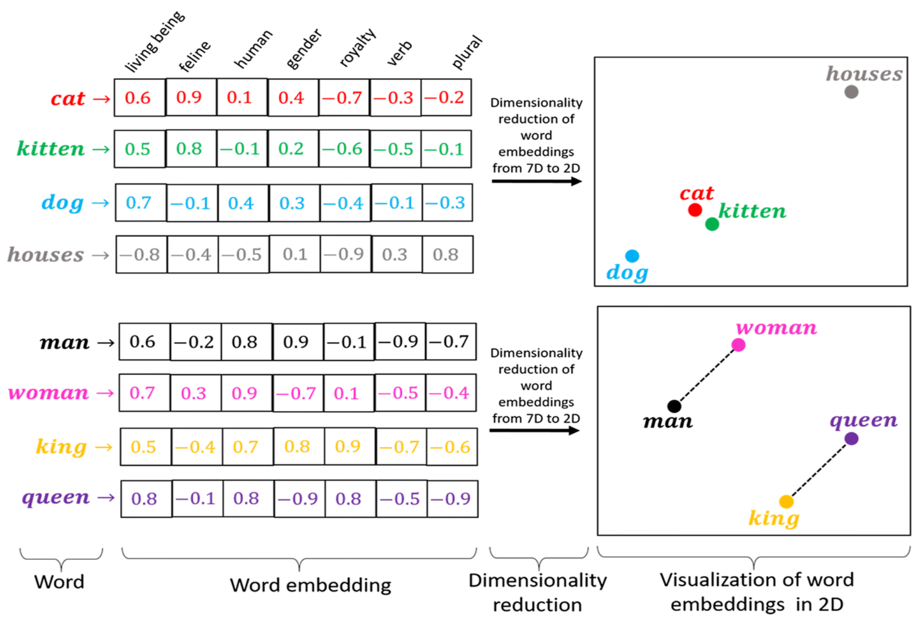
living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

FIGURE 3.1: Word2Vec embedding.

3.4.3 Supervised Learning with Sliding Windows

Sliding window models convert the next-in-sequence prediction problem into a traditional supervised learning problem that can be solved with any classifier such as decision trees, feed-forward neural networks and learning-to-rank methods. The general idea of the approach, which resembles auto-regressive models, is as follows. A sliding window of size W is moved over each sequence (see Figure 3.2). At each step, all items within the window are used to derive the feature values of the supervised learning problem and the identifier of the immediately next item is used as target variable. As a result, the sequence prediction problem is turned into a multi-class classification problem, or into a multi-label classification in case multiple target items are allowed [20].

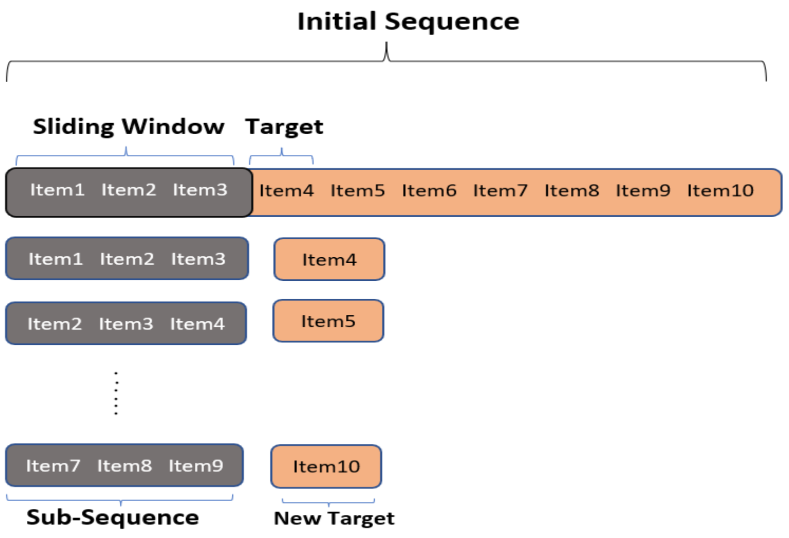

FIGURE 3.2: supervised learning with sliding windows.

3.4.4 Network architecture

The general neural network architecture can be seen in Figure 3.3. Considering a sequence with t interactions, the first step taken by the network is to convert each of the items in the t interactions to their corresponding embedding by taking them through an embedding layer. For embeddings of dimensionality d, the input to the first convolutional layer is of shape $t \times d$. Stride is taken as 1 and padding is set so that the spatial output dimensionality is also t. d kernels are applied in each layer to produce d stacked activation maps, making the final output of each convolutional layer $t \times d$. By using convolutions and retaining the spatial dimensionality, the final convolutional layer produces a d-dimensional output embedding for each time step in the sequence, $\mu_1, ..., \mu_t$. The output embedding used to make a prediction over the whole sequence is thus in the last slice t, where all interactions have been seen by the network. To make the analogy simple two types of layers are proposed in compact 1D CNNs: 1) the so-called "CNNlayers" where both 1D convolutions and sub-sampling (pooling) occur, and 2) Fully-connected layers that are identical to the layers of a typical Multi-layer Perceptron (MLP) and hence called as "MLP-layers". The configuration of a 1D-CNN is determined by the following hyper-parameters:

- Number of hidden CNN and MLP layers/neurons.
- Filter (kernel) size in each CNN layer.
- Subsampling factor in each CNN layer.
- The choice of pooling and activation operators.

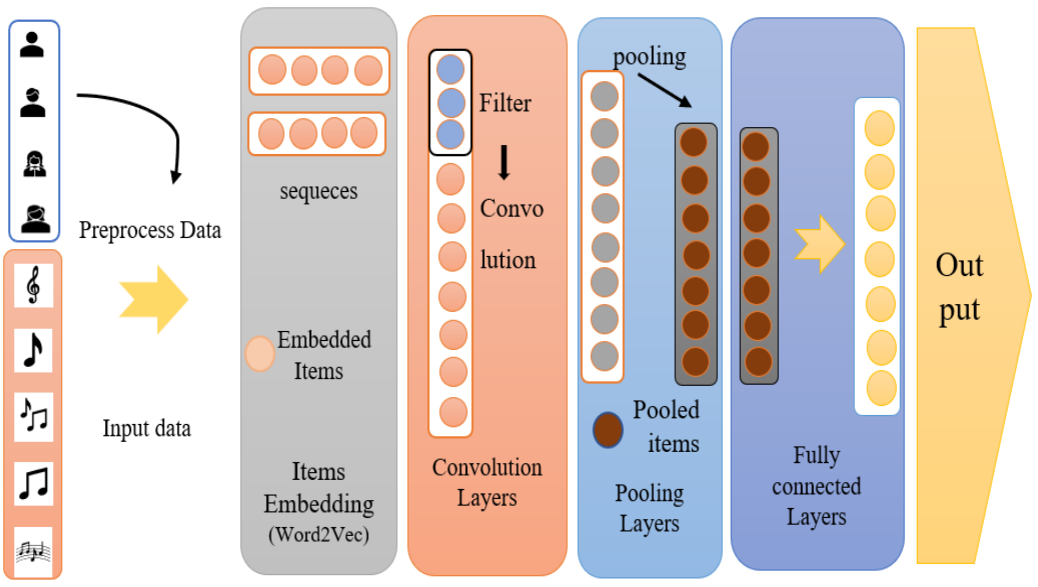

Figure 3.3: Architecture of convolutional neural network for sequential recommender system.

3.5 Experiments

3.5.1 Last.fm Dataset

What is Last.fm Dataset[1]?

This dataset contains <user, timestamp, artist, song> tuples collected from Last.fm API, using the user.getRecentTracks() method.

This dataset represents the whole listening habits (till May, 5th 2009) for nearly 1,000 users.

Data Statistics:

Total Lines:	19,150,868
Unique Users:	992

Data Format:

The data is formatted one entry per line as follows (tab separated, "\t"):

userid-timestamp-artid-artname-traid-traname.tsv

userid \t timestamp \t musicbrainz-artist-id \t artist-name \t musicbrainz-track-id \t track-name

Splitting the Data:

The dataset is comprised of items viewed by users. These views were converted to sequences by sorting the interactions of each user by time. The user-item views were split into a train set 90% and a test set 10%. The split is time-based, meaning that any interaction in the test set is at least as late as the latest interaction in the train set. The advantage of this split is that it reassembles a more realistic setting than randomly splitting user-item views which would break the sequential nature of the data(see Figure 3.4). Further, sequences of length less than 2 were removed from the dataset since predictions can not be made from empty sequences [51].

3.5.2 Evaluation Metrics

Recommendation qualities are commonly expressed through a number of metrics and methods. The choice of these is often based on the type of dataset used in the system, the use case, expected outcome, etc. Arguably the most common metric in recommender systems (and information retrieval) is the precision (P) and recall (R) pair [Her+04]. These metrics are usually applied in offline train/test scenarios, where algorithms are trained using a portion of the available data and then evaluated by comparing predictions to a withheld portion of the data, i.e. true positive recommendation.

Precision is the fraction of relevant retrieved documents. In a recommender system evaluation setting it corresponds to the true positive fraction of recommended items.

[1]http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html

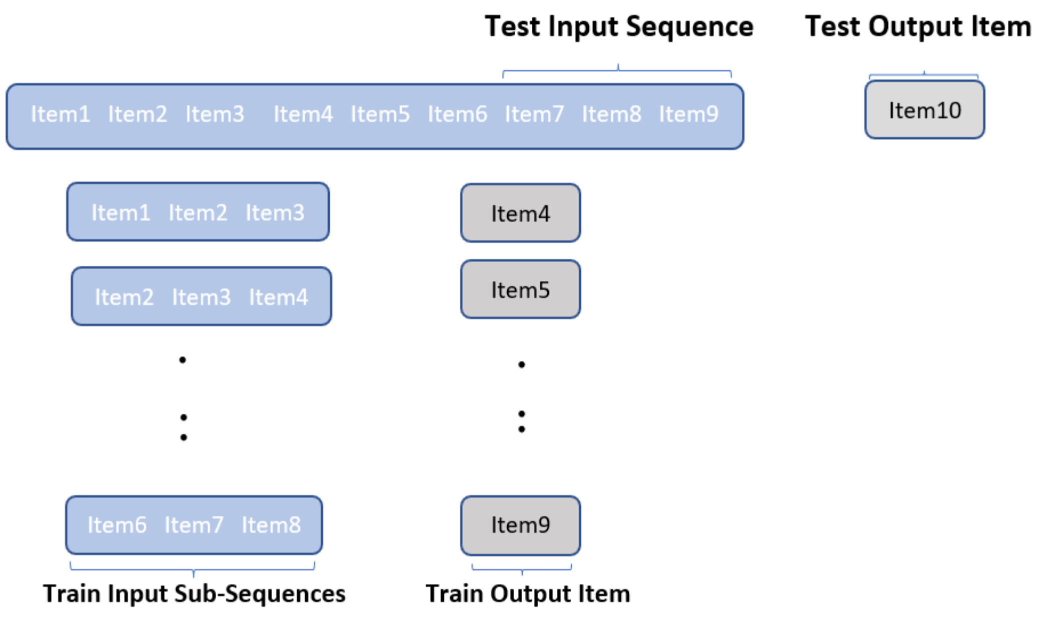

FIGURE 3.4: Splitting data technique.

Recall is the fraction of all relevant items which are retrieved. The formula for calculating precision is shown in Eq.(3.3) while recall is shown in Eq.(3.2). In both equations, relevant refers to the complete set of relevant items, and retrieved refers to the complete set of retrieved items.

$$P = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{retrieved\}|} \tag{3.1}$$

$$R = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{relevant\}|} \tag{3.2}$$

Commonly, precision is expressed as precision at k where k is the length of the list of recommended items, e.g. P@1 $= 1$ would indicate that one item was recommended, and the item was deemed to be a true positive recommendation, P@2 $= 0.5$ would indicate that two items were recommended and one them was deemed a true positive, etc [52]. Another commonly used measure is F-measure that is defined as:

$$F - measure = \frac{2.Precision.Recall}{Precision + Recall} \tag{3.3}$$

3.5.3 Implementation

The implementation was done in Python 3.7 with the jupyter notebook. And the neural networks were implemented in Keras.

3.5.3.1 Python

Python[2] 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.

FIGURE 3.5: Python programming language logo.

3.5.3.2 Jupyter Notebook

The Jupyter Notebook[3] is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

FIGURE 3.6: Jupyter notebook logo.

[2]https://www.python.org/
[3]https://jupyter.org/

3.5.3.3 Keras

Keras[4] is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

FIGURE 3.7: Keras library logo.

3.5.4 Results

Figure 3.8 shows a scatter plot that provide clear comparisons between the test values (test items) and the values predicted by the proposed CNN model (predicted items).

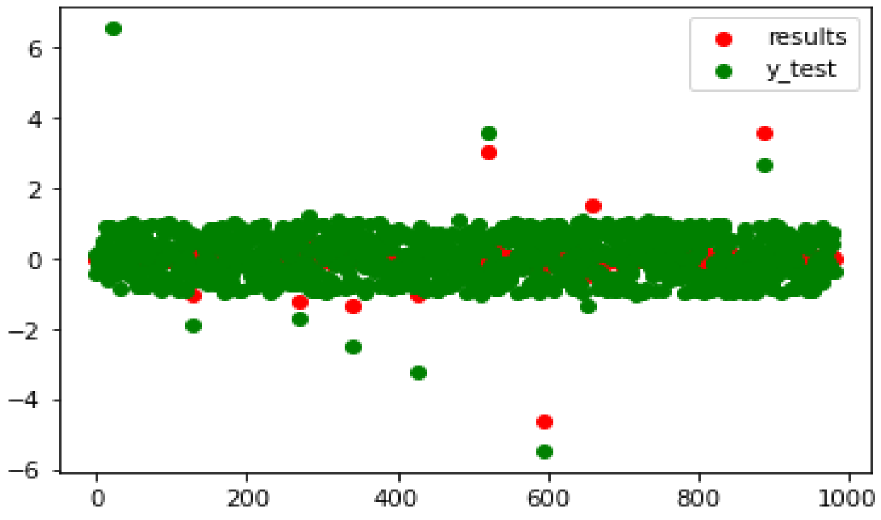

FIGURE 3.8: Scatter plot of test items and predicted items.

Table 3.1 shows the results of the proposed CNN model on the Last.fm Dataset. Several architectures were examined and a single layer of Convolution was found to be the best performer. adding additional layers always resulted in worse performance, the exact reason of this is still unknown and requires further research.

[4]https://keras.io/

Results			
Embedding size:1		Embedding size:10	
1 convolution layer	2 convolution layers	1 convolution layer	2 convolution layers
Precision: 0.0345 Recall: 0.5183 F-Mesure: 0,0646	Precision: 0.0210 Recall: 0.4802 F-Mesure: 0,0402	Precision: 0.0254 Recall: 0.3305 F-Mesure: 0,0471	Precision: 0,0338 Recall: 0.4403 F-Mesure: 0,0627

TABLE 3.1: Convolutional neural network results.

3.6 Conclusion

In this chapter, we have proposed a convolutional neural network model to make sequential recommendations. The (Word2Vec) is further introduced to produce items embedding. We also touched on a number of ways to evaluate the performance of recommender systems to show the importance of using a deep learning model to make sequential recommendation.

Conclusion

A. Summary

Sequential Recommendation is a highly relevant problem in practice. Researchers have developed a variety of algorithmic proposals over the past fifteen years. In the first chapter of this thesis we present types of recommender systems, traditional recommenders systems challenges and general principles of empirical research and current state of practice in sequential recommendation techniques were presented.

In Chapter 2, deep learning was investigated in depth from fundamentals, i.e. neural networks, learning algorithms, activation functions, for the purpose of adopting a deep learning approach which is convolutional neural networks in sequential recommendations.

To answer the research questions, a sequential recommender system was designed, built, and tested in chapter 3. The model provides recommendations with a convolutional neural network.

The thesis is concluded by reflecting back to the research question. The research question was:

"*How can convolutional neural networks be used in sequential recommender systems*?"

With our approach, we showed how convolutional neural networks can be used to incorporate sequential data in recommender systems.

B. Directions for future research

The convolutional neural network architecture used in this thesis is relatively simple. It could be further extended on by including more advanced deep learning techniques such as dropout and weight normalization, shown successful in other sequence modelling domains. Adding more advanced techniques could further improve the predictive performance of the network.

Whilst the current network only considers sequences with fixed length, the network could also be extended to incorporate sequences with different length and compare results to each other.

In the future, more experiments can be done using other datasets used in the recommendation literature and Compare the model to a set of commonly used baselines to observe the effectiveness of the proposed method.

Although theory suggests that convolutional networks are faster than recurrent networks, empirically evaluating the time performance for sequential recommendations during both training and prediction would be an interesting study and could further motivate the use of convolutional networks instead of recurrent networks in time-critical domains such as recommender systems.

Bibliography

[1] M Sándor Apáthy. Thesis of dissertation.

[2] LIU Liling. Summary of recommendation system development. In *Journal of Physics: Conference Series*, volume 1187, page 052044. IOP Publishing, 2019.

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

[4] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[5] Joeran Beel, Stefan Langer, Marcel Genzmehr, Bela Gipp, Corinna Breitinger, and Andreas Nürnberger. Research paper recommender system evaluation: a quantitative literature survey. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, pages 15–22, 2013.

[6] Shoujin Wang, Gabriella Pasi, Liang Hu, and Longbing Cao. The era of intelligent recommendation: Editorial on intelligent recommendation with advanced ai and learning. *IEEE Annals of the History of Computing*, 35(05):3–6, 2020.

[7] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134, 2002.

[8] Folasade Olubusola Isinkaye, YO Folajimi, and Bolande Adefowoke Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.

[9] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[10] Abhaya Kumar Sahoo, Sitikantha Mallik, Chittaranjan Pradhan, Bhabani Shankar Prasad Mishra, Rabindra Kumar Barik, and Himansu Das. Intelligence-based health recommendation system using big data analytics. In *Big data analytics for intelligent healthcare management*, pages 227–246. Elsevier, 2019.

[11] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems*, pages 47–51, 2010.

[12] Frédéric Guillou. *On recommendation systems in a sequential context.* PhD thesis, Université Lille 3, 2016.

[13] Zeshan Fayyaz, Mahsa Ebrahimian, Dina Nawara, Ahmed Ibrahim, and Rasha Kashef. Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *Applied Sciences*, 10(21):7748, 2020.

[14] Pradeep Kumar Singh, Pijush Kanti Dutta Pramanik, Avick Kumar Dey, and Prasenjit Choudhury. Recommender systems: an overview, research trends, and future directions. *International Journal of Business and Systems Research*, 15(1): 14–52, 2021.

[15] Gilbert Ritschard. Exploring sequential data. In *International Conference on Discovery Science*, pages 3–6. Springer, 2012.

[16] Hui Fang, Guibing Guo, Danning Zhang, and Yiheng Shu. Deep learning-based sequential recommender systems: Concepts, algorithms, and evaluations. In *International Conference on Web Engineering*, pages 574–577. Springer, 2019.

[17] Massimo Quadrana. Algorithms for sequence-aware recommender systems. 2017.

[18] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830*, 2019.

[19] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. Deep learning for sequential recommendation: Algorithms. *Influential Factors, and Evaluations. arXiv: Information Retrieval URL https://arxiv. org/abs*, 1905.

[20] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.

[21] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019.

[22] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Convolutional sequence modeling revisited. 2018.

[23] Francois Chollet et al. *Deep learning with Python*, volume 361. Manning New York, 2018.

[24] Hans-Dieter Wehle. Machine learning, deep learning, and ai: What's the difference? In *International Conference on Data scientist innovation day, Bruxelles, Belgium*, 2017.

[25] Ciro Donalek. Supervised and unsupervised learning. In *Astronomy Colloquia. USA*, volume 27, 2011.

[26] Asad Abdi. Three types of machine learning algorithms, 2016.

[27] Nur Farhana Hordri, Siti Sophiayati Yuhaniz, and Siti Mariyam Shamsuddin. Deep learning and its applications: a review. In *Conference on Postgraduate Annual Research on Informatics Seminar*, 2016.

[28] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

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

[30] Agnieszka Bojanowska and Marek Milosz. Application of neural networks in crm systems. In *ITM Web of Conferences*, volume 15, page 04001. EDP Sciences, 2017.

[31] Jason Brownlee. *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery, 2016.

[32] Crescenzio Gallo. Artificial neural networks tutorial. In *Encyclopedia of Information Science and Technology, Third Edition*, pages 6369–6378. IGI Global, 2015.

[33] Carlos Gershenson. Artificial neural networks for beginners. *arXiv preprint cs/0308031*, 2003.

[34] Luis A Camuñas-Mesa, Bernabé Linares-Barranco, and Teresa Serrano-Gotarredona. Neuromorphic spiking neural networks and their memristor-cmos hardware implementations. *Materials*, 12(17):2745, 2019.

[35] Roger Grosse. Lecture 5: Multilayer perceptrons. *inf. téc*, 2019.

[36] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019.

[37] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[38] Xiao Dong, Jiasong Wu, and Ling Zhou. How deep learning works–the geometry of deep learning. *arXiv preprint arXiv:1710.10784*, 2017.

[39] Jeff Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning, 2018.

[40] Anirudha Ghosh, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, and Debashis De. Fundamental concepts of convolutional neural network. In *Recent Trends and Advances in Artificial Intelligence and Internet of Things*, pages 519–567. Springer, 2020.

[41] Gil Keren and Björn Schuller. Convolutional rnn: an enhanced model for extracting features from sequential data. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3412–3419. IEEE, 2016.

[42] Francois Chollet. *Deep learning with Python.* Simon and Schuster, 2017.

[43] Nebojša Bačanin Džakula et al. Convolutional neural network layers and architectures. In *Sinteza 2019-International Scientific Conference on Information Technology and Data Related Research*, pages 445–451. Singidunum University, 2019.

[44] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach.* " O'Reilly Media, Inc.", 2017.

[45] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5:23, 2017.

[46] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. 2020. *URL https://d2l. ai*, 2020.

[47] Sagar Sharma and Simone Sharma. Activation functions in neural networks. *Towards Data Science*, 6(12):310–316, 2017.

[48] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. Hierarchical temporal convolutional networks for dynamic recommender systems. In *The world wide web conference*, pages 2236–2246, 2019.

[49] Jason Brownlee. *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python.* Machine Learning Mastery, 2018.

[50] Makbule Gulcin Ozsoy. From word embeddings to item recommendation. *arXiv preprint arXiv:1601.01356*, 2016.

[51] Tim Kerschbaumer. Convolutional neural networks for sequence-aware recommender systems. Master's thesis, 2018.

[52] Alan Said. Evaluating the accuracy and utility of recommender systems. 2013.