REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET DE l'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de **Master**

Spécialité : Génie Logiciel

En vue de créer une startup

KHADAMNI.DZ

Par :

TOUHAMI Hadil
YAHIAOUI Somia

Sur le thème

Session-based Recommendation Systems with Graph ATtention Networks

Soutenu publiquement le 11 / 07 / 2023 à Tiaret devant le jury composé de :

Mr. KOUADRIA Abderrahmane	MCB	Université de Tiaret	Président
Mr. BOUDAA Boudjemaa	MCA	Université de Tiaret	Encadrant
Mr. BERBER EL-Mehdi	MAA	Université de Tiaret	Examinateur
Mme. SADJI Fatima	Pr	Université de Tiaret	Représentant de l'incubateur
Mr. REKHIES Mohamed		ANEM de Tiaret	Repésentant du partenaire socio-économique

2022-2023

Acknowledgments

First and foremost, we express our deepest gratitude to Allah, the Almighty, for guiding us throughout this research journey and granting us the strength and perseverance to complete this thesis.

We would like to extend our sincere appreciation to our supervisor, BOUDAA Boudjemaa, for their invaluable guidance, continuous support, and constructive feedback. Their expertise and dedication played a vital role in shaping this work.

We are also indebted to our parents, sisters, brothers, and extended family members for their unwavering love, encouragement, and belief in our abilities. Their constant encouragement and sacrifices have been instrumental in our academic pursuits.

Furthermore, we would like to express our heartfelt gratitude to our friends, whose unwavering support and encouragement have been a constant source of motivation. Their presence during both the challenging and joyous moments has made this journey more meaningful and enjoyable.

We extend our gratitude to all individuals, whether directly or indirectly involved, who have contributed to our research. Their assistance, whether in the form of discussions, interviews, or provision of resources, has greatly enriched this work.

Lastly, we acknowledge the efforts of all the researchers, scholars, and pioneers in our field of study, whose previous work has served as a foundation and inspiration for our own.

We are deeply grateful to all the mentioned individuals for their invaluable contributions and support. However, any omissions of names are unintentional, and we appreciate the understanding of those who may not be mentioned explicitly.

Thank you all for being an integral part of our journey and for shaping us into the individuals we have become today.

Dedication

Dedication

To our loving parents Mohamed, Fatima, Ayachi and Fatma (Tamtam),

supportive siblings Abedselam, Oussama, Habib, Djihad, Kadirou, Sara, Hadjer and Tasnim,

dear friends Widad, Sabrine, and cherished family,

This thesis is dedicated to each and every one of you. Your unwavering love, encouragement, and belief in us have been instrumental in our academic journey. Your presence, guidance, and support have shaped our accomplishments and inspired us to reach new heights.

Thank you for being our constant source of strength, for celebrating our successes, and for standing by us during challenging times. This thesis is a testament to the profound impact you have had on our lives.

With heartfelt gratitude and sincere thanks

TOUHAMI Hadil & YAHIAOUI Somia

Table of contents

Contents

Abstract

Recommender systems give users beneficial product or service recommendations for their decision-making processes. Today, a variety of application domains such as YouTube, Amazon, Facebook, and ResearchGate have proven the validity of classic recommendation systems that employ collaborative and content-based filtering techniques. Session-based recommender systems, a novel RS paradigm, have evolved in recent years in order to give timelier and more accurate next-item recommendations that are responsive to being adjusted in various session circumstances. SBRSS strives to record dynamic and short-term user preferences within sessions. The literature only includes a few models with poor precision and efficacy as proposed development methodologies for SBRS models, which are this type of system's primary objectives. The goal of this thesis is to explicitly provide a new deep learning design for session-based recommender systems based on graph neural networks (GNNs) via the intriguing architecture of graph attention networks (GAT). Currently, gat-based methodologies are among the most cutting-edge techniques used in many research fields, and SBRs can take advantage of them to greatly research fields, and SBRs can take advantage of them to greatly enhance the outcomes of their suggestions.

Keywords: recommender systems, session-based recommender system, graph neural network, graph attention network.

Résumé

Les systèmes de recommandation donnent aux utilisateurs des recommandations de produits ou de services bénéfiques pour leurs processus de prise de décision. Aujourd'hui, une variété de domaines d'applications tels que YouTube, Amazon, Facebook et ResearchGate ont prouvé la validité des systèmes de recommandation classiques qui utilisent des techniques de filtration collaboratives et basées sur le contenu. Les systèmes de recommandation basés sur la session, un nouveau paradigme RS, ont évolué au cours des dernières années afin de fournir des recommandations plus ponctuelles et plus précises qui répondent à des ajustements dans diverses circonstances de la session. SBRSS s'efforce d'enregistrer les préférences d'utilisateur dynamiques et à court terme au sein des sessions. La littérature ne comprend que quelques modèles de faible précision et efficacité en tant que méthodologies de développement proposées pour les Modèles SBRS, qui sont les objectifs principaux de ce type de système. L'objectif de cette thèse est de fournir explicitement une nouvelle conception d'apprentissage profond pour les systèmes de recommandation basés sur la session basée sur les réseaux neuronaux graphiques (GNNs) via l'architecture intrigante des réseaux d'attention graphique (GAT). Actuellement, les méthodologies basées sur le trou sont parmi les techniques les plus avancées utilisées dans de nombreux domaines de recherche, et les SBR peuvent en tirer profit dans des domaines très recherchés, et ils peuvent en profiter pour améliorer considérablement les résultats de leurs suggestions.

Mots clés : système de recommandation, système de recommandation basé sur la session, réseaux neuronaux graphiques, réseau d'attention graphique.

List of figures

List of tables

Acronyms

RS	**R**ecommender **S**ystems
SBRS	**S**ession-**B**ased **R**ecommendation **S**ystems
GNN	**G**raph **N**eural **N**etworks
GAT	**G**raph **AT**tention Networks
CBF	**C**olla**b**orative **F**iltering
ML	**M**achine **L**earning
DL	**D**eep **L**earning
SARS	**S**ession-**A**ware **R**ecommendation **S**ystems

General Introduction

General introduction

1. Background

Since the first articles on collaborative filtering appeared in the middle of the 1990s, recommender systems have gained importance as a study area [1]. The distinctive features of recommender systems have undergone extensive development over the past few decades in both industry and academia. There are several pertinent apps that can assist users in managing information overload and offer them individualized recommendations, content, and services, but there are still a lot of unanswered questions in this field of study. We now routinely receive different types of automated recommendations when using the internet. Currently, numerous application areas, including e-commerce and video streaming, use these platforms.

The matrix completeness problem formulation, where a user-item rating matrix is given and the goal is to forecast the missing values, has historically functioned as a standard framework for scholarly research in the area. The training of machine learning models that aim to capture longer-term user preference profiles is generally well suited to this abstraction. Sequence-aware recommender systems, on the other hand, use additional methods to execute recommendations that take into consideration users' near-term actions and intentions. They are also designed to benefit from the rich data that is contained in the sequentially organized user interaction logs that are typically seen in real-world applications.

Recommender systems are critical in many application scenarios where short-term user interests and longer-term sequential patterns are vital to their success. Session-based recommender systems are a typical example, where there is no longer-term user history available and the recommendations must be adapted based on the assumed short-term interests of an anonymous user. The primary goal in such scenarios is to recommend objects that match a given sequence of user actions.

Since the 1990s, a lot of research has focused on session-based recommender systems, which have also been referred to as next-item, next-basket, pattern-based, rule-based, sequence-based, and transaction-based recommender systems [2].

The pertinent works on session-based recommender systems are divided into two distinct stages: the model-free stage from the late 1990s to the early 2010s, and the model-based stage from the early 2010s to the present. To drive the model-free stage, data mining techniques such as pattern mining, association rule discovery, and sequence mining were developed. Research on recommendations based on patterns, rules, and sequences eventually took control graph-based machine learning techniques, particularly graph neural network (GNN) models. Model-based recommender systems have advanced to new heights since 2017 as a result of the recent explosive expansion of GNNs. Researchers have just started working in this area and have developed a variety of GNN-based models for next-item and shopping cart predictions.

Due to its outstanding performance over the past few decades in numerous application areas, including computer vision and speech recognition, deep learning is currently the subject of a lot of excitement. Academics and business are vying to incorporate deep learning into a wider range of applications due to its capacity to handle numerous complicated problems.

Recent developments in deep learning have changed recommendation architectures and expanded opportunities for increasing recommender system performance [3]. Graph neural networks (GNNs), which have been proved to be particularly good at processing data that is graph-structured, are the foundation for the majority of current deep learning models. Due to their capacity to record intricate patterns in user-element interactions and their capacity for generalization, GNs have consequently drawn considerable interest in session-based recommender systems. The effectiveness of session-based recommender systems could be greatly increased by GNNs. The interpretability of the recommendations has also been improved by the addition of a GNNS attention mechanism.

2. Problem Statement

Session-based recommendation systems (SBRS) have grown in popularity in recent years as businesses attempt to provide customized recommendations to users based on their interactions with items in the short term.

Traditional SBRS methods, on the other hand, frequently rely on basic models that fail to capture the complicated and dynamic patterns of user behavior that emerge within individual sessions, resulting in poor suggestions.

To solve this problem, this thesis proposes a Graph Attention Network (GAT) model for SBRS that is capable of capturing the complex interactions between items within a session and creating correct and appropriate recommendations for users. This effort is guided by a central research question:

“"How can a Graph Attention Network (GAT) model be effectively utilized in session-based recommendation systems to capture the intricate and dynamic patterns of user behavior within individual sessions, leading to improved and personalized recommendations?"

By addressing this issue, this study intends to investigate the potential of GAT models for boosting SBRS performance and promoting user happiness across a variety of sectors, including e-commerce, news, and entertainment.

3. Delimitation

This thesis is limited to the examination of item-item and item-session interactions and focuses on session-based recommendation systems employing the GAT model. The effectiveness of the GAT model will be assessed using a publicly accessible dataset, and the study will not take into account how external variables like user demographics, temporal dynamics, or contextual information may affect the quality of the recommendations. Additionally, this study excludes other domains like news

and music in favor of evaluating the GAT model's performance just n the e-commerce domain. This thesis intends to provide a clear and focused study of the GAT-based SBRS model's performance on a specific dataset and in a specific domain by outlining these constraints. noting the need for additional research in many fields and under various experimental setups as well as the potential influence of other factors.

4. Approach

In order to establish a session-based recommendation system using graph neural networks (GNNs), specifically the Graph Attention Network (GAT) model, The main problem of the research is the need for a system that can produce personalized recommendations based on user experiences while utilizing the wide range of information found in the session graph.

Each item clicked during a browser session is treated as a node in the session graph, which depicts the user's sequential interactions. The GAT model will be used to capture the intricate dependencies and relationships between these nodes, allowing the system to provide recommendations that are aware of their context.

All in all, it is expected to offer a strong framework for comprehending user preferences, recognizing similar items, and making precise recommendations by customizing the GAT model to fit the session-based recommendation requirement.

5. Outline

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

- *Introduction:* initiation to recommender systems and the background, problem statement, delimitations of the thesis and the approach.
- *Chapter 1: Session-Based Recommendation Systems*
  A theoretical introduction to recommender systems and their families, all for the purpose of giving a basic foundation and a comprehensive view of session-based recommender systems.
- *Chapter 2: Graph ATtention Networks*
  Essential context and background knowledge around artificial intelligence, machine learning, and deep learning. This chapter gives a familiarization with the fundamental concepts necessary to approach deep learning, GNNs and GATs.
- *Chapter 3: Graph ATtention Networks for the development of Session-based recommendations systems*
  Takes an in-depth dive into practical application of graph attention networks in the making of a session-based recommendation systems. The evaluation metrics used for the experiment are first presented, then datasets and baselines used. At last, the baselines results are compared to the proposed GAT-based SBRS model with a discussion of the obtained results.
- *Conclusion:* Based on the results and the discussion in the previous chapter, this last part relates to the research questions and draws a summary of this work. Finally, suggestions for potential future work are discussed.

Chapter1.

Session-based recommendation systems

Chapter 1 : Session-Based Recommendation Systems

1.1 Introduction

Recommendation systems have become essential tools for assisting users in navigating the vast array of services and products accessible in the era of many options and overwhelming of information. These systems aim to offer recommendations that are tailored to each user's interests , leading to more informed decision-making.

The efficacy of conventional recommendation systems is undeniable, yet they frequently fail to capture the changing nature of user preferences. They ignore a user's short-term transactional behaviors and potential preference changes over time, focusing instead on their long-term, unchanging preferences. This constraint necessitates a novel strategy that considers the temporal component of user behavior.

Session-based recommendation systems are useful in this situation. In recent years, session-based recommender systems have drawn a lot of attention since they provide a relatively new paradigm for recommendation since they provide a relatively new paradigm for recommendation. By taking into account the context of a user's current session or visit and including short-term transactional patterns, these systems go beyond static preferences.

In this chapter, we will delve into the world of session-based recommender systems, offering a comprehensive and systematic overview of this emerging recommendation approach. We will explore the key concepts and techniques employed in session-based recommendation systems, shedding light on how they address the limitations of traditional recommendation systems.

1.2 Recommendation Systems

A recommender system, usually referred to as a recommendation system, is a method of making decisions that is intended to help users in complicated information settings, especially in the context of E-commerce [4].

It acts as a tool that facilitates users' effective searching through a vast amount of knowledge tailored to their preferences and areas of interest [5]. The system leverages recommendations from others to augment the decision-making process when users lack personal knowledge or experience with the available alternatives [6]. By offering individualized and unique recommendations for content and services, it tackles the issue of information overload [7].

In essence, a recommender system makes use of artificial intelligence algorithms and techniques to analyze user data, such as past purchases, search histories, demographic data, and more, in order to suggest or recommend more products, information, or services that are most relevant to a specific user [8]. It serves as a tactical tool that helps users identify pertinent information, improves decision-making by incorporating social recommendations, and reduces information overload by providing tailored and specialized recommendations.

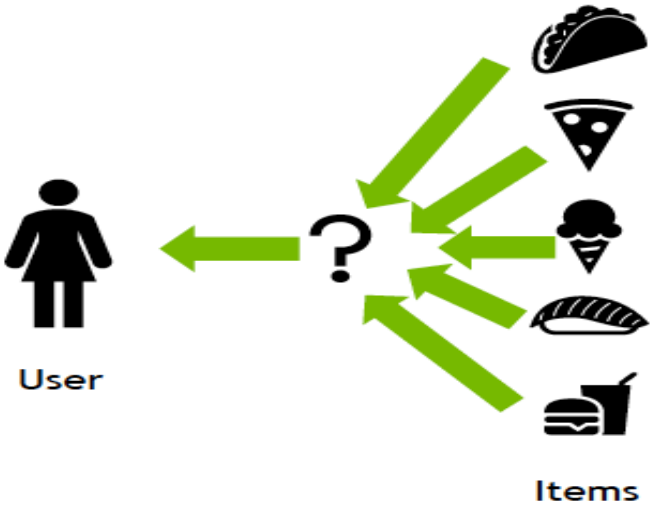


Figure 1 : Example of recommendation system

1.3 User Feedback

It is important to define the concept of user feedback, which is a key concept of recommendation. When users perform actions over items on a platform (e.g., an e-commerce site, a movie review system , a video blogging platform , and so on) , they provide implicit, explicit or , hybrid feedback:

1.3.1 Explicit Feedback

In order to build and enhance his model, the system typically prompts the user through the system interface to offer ratings for items. The quantity of user-provided ratings determines how accurate the recommendation is. This method's only shortcoming is that users must put forth some effort and aren't always ready to provide sufficient data. Since it does not involve deriving preferences from actions and it provides transparency into the recommendation process, explicit feedback is still perceived as providing more reliable data, leading to a marginally higher perceived recommendation quality and greater confidence in the recommendations [9].

1.3.2 Implicit Feedback

By keeping track of a user's many actions, including past purchases, navigation history, time spent on specific web pages, links they follow, email content, and button presses, among other things, the system automatically infers what the user prefers. By assuming users' preferences based on how they interact with the system, implicit feedback lessens the burden on users. Although the method does not require human effort, it is less accurate. Additionally, it has been argued that implicit preference data may actually be more objective because there is no bias due to users answering in a socially acceptable manner and because there are no concerns with self-image or the desire to maintain an image for others [10].

1.3.3 Hybrid Feedback

The strengths of both implicit and explicit feedback can be combined in a hybrid system in order to minimize their weaknesses and get a best performing system. This can be achieved by using an implicit data as a check on explicit rating or allowing user to give explicit feedback only when he chooses to express explicit interest [11].

Many recommender systems focus on implicit input since it is simple to collect and represent users' opinions through observable behavior (e.g., by examining browsing history, mouse movements, etc.).

Implicit feedback has the drawback that it is always noisy. For instance, even if a person has viewed a movie, we can't say for sure if they enjoyed it or not. However, it is not always available because some users are unwilling to score the products they consume. Explicit feedback, on the other hand, might be more illuminating about user preferences [12].

1.4 Recommendation tasks

We can specify many types of recommendation tasks that a recommendation system can take on depending on the type of feedback that is available [12]:

- Click-through rate (CTR) prediction*:* Predicting the probability that a user would click on an item that is described by a set of features (such as an image, text, the day of the week, a user's features, etc.), namely predicting P (click|item, user, features) of implicit feedback occurring.
- Rating prediction: estimating the probability that a user will give a particular rating to a product that has a particular collection of attributes, specifically estimating the P (Rating=r|item, user, features) of giving a specific product a rating r.
- Sequential prediction: determining the probability distribution of the next target item that a user would consume based on the attributes of the previous target item in a sequence. In addition to their IDs, both the user and the item sequence could be described by a different set of characteristics. If user features are not taken into account, the issue results in a broad (as opposed to customized) recommendation.

1.5 Types of Recommendation Systems

Although there are many recommendation algorithms and strategies, the majority can be divided into the following categories:

1.5.1 Content-based Filtering

The information retrieval and filtering study is where the content-based recommendation algorithms are derived [13]. The early collaborative filtering process is continued and developed in the content-based recommendation. Instead of using the user's remarks on things, content-based recommendation systems suggest items that are similar to those that the user has previously selected. Many current content-based systems create profiles for both users and items. While an item profile contains a list of item features, a user profile comprises details about a user's tastes, preferences, and needs that can be extracted from questionnaires about users or over time from their transactional behavior. The system then determines the degree to which each item's profile and the user's profile match up, recommending products that might satisfy the user's needs or preferences. The utility function is typically defined as the following by combining the features of items and the user interest model [14]:

1. $u(c,s)=score\big(ContentBaseProfile(c), Content(s)\big)$

u(c, s): This represents the utility or score of a content item (s) for a specific user (c). It predicts how much the user will like or find the content item relevant.

ContentBaseProfile(c): This refers to the profile of the user (c) in the content-based recommendation system. The profile consists of attributes or features that describe the user's preferences or interests.

Content(s): This represents the content item (s) that is being evaluated for recommendation. It is described by attributes or features that capture its characteristics.

score(ContentBaseProfile(c), Content(s)): This function calculates the similarity or relevance score between the user's profile and the content item. It measures how well the attributes of the content item align with the user's preferences or interests

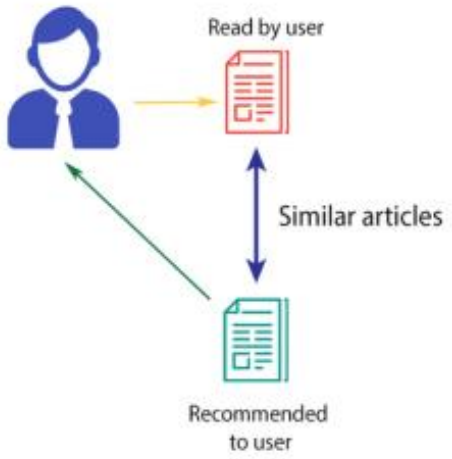


Figure 2 : Example of content-based filtering

1.5.2 Collaborative filtering

According to user preferences collected from a large number of users, algorithms recommend products (this is the filtering component). Using the similarity of user preference behavior and knowledge of past interactions between users and objects (this is the collaborative part), recommender systems can learn to anticipate future interactions. These recommender systems create a model from a user's prior actions, such as items they have previously purchased, ratings they have given for those items, and comparable choices made by other users. According to the theory, there is a good chance that two people will choose the same things in the future if they have already made comparable decisions and purchases in the past, such as choosing a movie [7].

For instance, if a collaborative filtering recommender discovers that you and another user have similar movie preferences, it can suggest a film to you that it has previously discovered that the other user enjoys (see figure 3) [15].

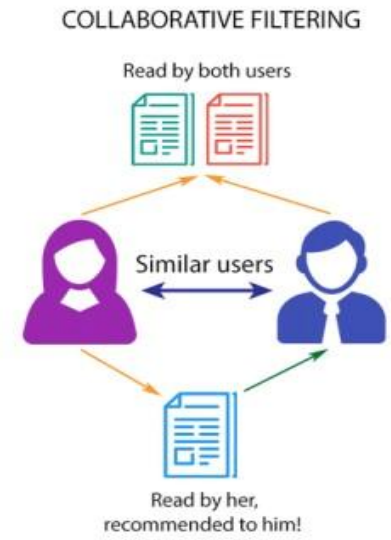


Figure 3 : Example of collaborative filtering

1.5.3 Hybrid Filtering

A hybrid recommendation system is a special sort of recommender system that gives the user a recommendation by combining two or more methods, such as collaborative and content-based filtering methods. The difficulties posed by employing these two filtering techniques independently were overcome by combining them [11].

Based on how the various recommended methodologies are combined with one another, hybrid RSs are divided into 7 classes [16]:

- Weighted***:*** We can define a few models that can accurately understand the dataset for the weighted recommendation system. The weighted recommendation system will integrate the outputs from all of the models into static weightings, which remain the same throughout the training and test sets.
- Switching***:*** In a switching hybrid, the system alternates between different recommendation strategies based on a set of criteria, such as when a strategy doesn't generate enough reliable recommendations.
- Mixed: A mixed hybrid technique initially generates a variety of candidate datasets using the user profile and attributes. The recommendation system feeds various sets of candidates into the model in accordance with their requirement combining the predictions to produce the final recommendation.
- Feature Combination: This hybrid recommender system treats one recommender's output as extra feature data and prefers to employ the second recommender (often content-based, which heavily utilizes item features) over the new expanded data.
- Feature augmentation: A contributing recommendation model is used to grade or categorize the user/item profile. This rating , or categorization, is then used in the primary recommendation system to produce the expected outcome. Without altering the primary recommendation model, the feature augmentation hybrid can boost the performance of the core system. For instance, we can improve the user profile dataset by utilizing the association rule.

The functionality of the model for content-based recommendations will be enhanced by the augmented dataset.

- Cascade: One type of hybrid filtering method is cascade hybrids. A coarse ranking of the candidate items is first generated using one technique, and the list is subsequently refined using the preliminary candidate set. The cascades depend on the order.
- Meta Level: is also an example of order-sensitive hybrid RSs that use an entire model produced by the first technique as input for the second technique.

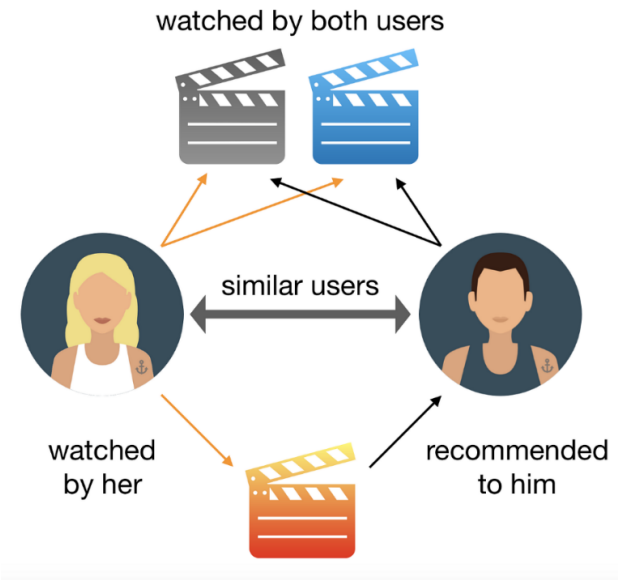


Figure 4 : Example of hybrid filtering

1.6 Challenges and Limitations

The following is a summary of the challenges and restrictions that RSs confront that are seen as crucial for the advancement of RSs research.

- Cold start problem: The system encounters the cold start issue when it is unable to establish any connections between users and items for which it lacks sufficient data. There are two different kinds of cold-start issues [17]:
  1. User cold-start issues: These issues occur when there is virtually no information available about the user.
  2. Product cold-start issues: These issues occur when there is virtually no information available about the product**.**
- Synonymy: When an item is represented by two or more names or entries that have similar meanings, synonymy results. In some situations, the recommender is unable to determine if the terms refer to separate or the same item [18].
- Shilling Attacks**:** When a dishonest user or rival enters a system and begins providing fraudulent ratings on certain items in an effort to either boost or diminish the item's popularity. Such attacks have the potential to undermine user confidence in the RS and harm the effectiveness and value of suggestions. CF techniques are more concerned with this threat [19].
- Privacy: Better recommendation services are obtained by providing personal information to the RS, but doing so may raise data privacy and security concerns.

- Problem of Overspecialization: Users are only shown recommended products based on those already known or defined by their user profiles, ignoring new items and other possibilities choices [18].
- The sparsity problem: occurs when a user has a large matrix of purchases, viewings, or music listings. It is a critical problem in recommender systems. When the user didn't rate these things, sparsity developed. While recommender systems rely on users ratings on a matrix to recommend items to others [18].
- Gray Sheep: Gray sheep happen in collaborative filtering systems when a user's thoughts do not align with any group, and as a result, they are unable to profit from recommendations [20].
- Novelty: New ones must be included in the recommended items.
- Serendipity: beyond novelty, it may also be an objective that some recommended items are not only unheard of, but also surprising user wouldn't have thought before [20].
- Scalability: is a measure of a system's capacity to operate efficiently with high performance as information expands. When the number of users or the number of items increases, the recommender system must continue to suggest the same items to the users. We need to perform more calculations and spend more money to achieve this [20].
- Evaluation and the Availability of Online Datasets: A recommender system's quality can be assessed along with other factors. One of the main issues with RSs is the construction of evaluation criteria and the selection of appropriate evaluation metrics. They can be categorized as follows [21]:
    1. Prediction metrics: include coverage and accuracy measures such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Normalized Mean Average Error (NMAE).
    2. Set recommendation metrics: such as Precision, Recall, and Receiver Operating Characteristic (ROC).
    3. Rank recommendation metrics: such as the half-life and the discounted cumulative gain.

1.7 Sequence-Aware Recommender Systems

A type of recommender systems known as sequence-aware recommender systems considers the chronological order of items in a user's interaction history. They are employed to anticipate the following items a user will find interesting during a current session or to generate complete sequences of items to display to the user [22].

Sequence-aware recommendation issues differ in several ways from conventional matrix-completion issues. The problem, its inputs, outputs, and specific computing tasks are all presented in high-level detail in Figure 5. In general, the sequence in which the items that presented can have an impact on both the inputs and the outputs. These points will be covered in more detail later on [23].

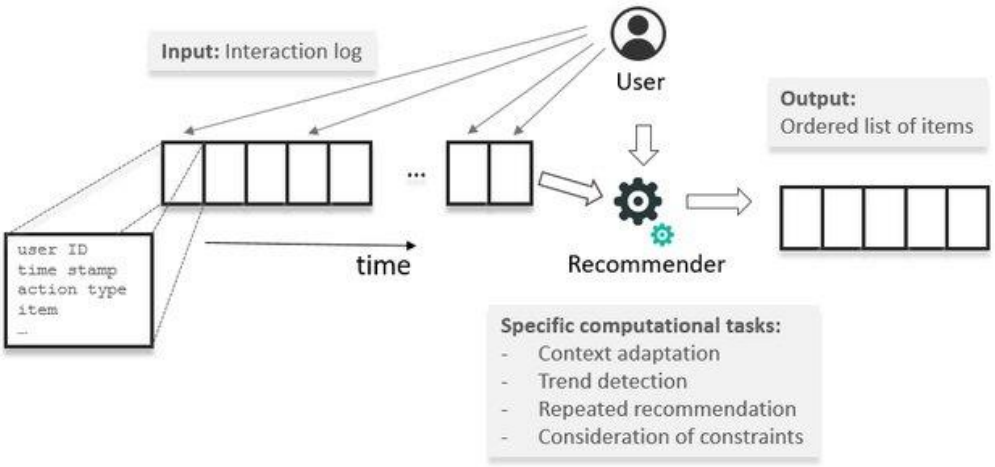


Figure 5 : Overview of sequence-Aware recommendation problem

1.7.1 Inputs : The primary input to sequence-aware recommendation problems is an ordered and frequently timestamped list of previous user actions. Users can be known to the system or be anonymous. Each action can also be one of several predefined types, and each action, user, and item may have a number of additional attributes [23].

1.7.2 Outputs : Ordered lists of items are the result of a sequence-aware recommender. The results in this broad sense resemble those of a conventional "item-ranking" recommendation arrangement. The order of the objects in the recommendation list, however, can also be important in some sequence-aware recommendations cases. There are situations where the user should take into account each recommendation and do so in the order supplied rather than viewing the list of recommendations as a collection of choices [23].

1.7.3 Computational Tasks : The literature has identified a variety of computational occupations for sequence-aware recommenders. Four primary objectives are most frequently accomplished with the use of SARS in a variety of application scenarios:

1) Context Adaptation***:*** One of the main objectives of sequence-aware recommender systems is to understand the scenario and aims of the users in order to make recommendations that are appropriately tailored to those situations. It is crucial to effectively utilize interactional context information since there is no historical data accessible regarding the users' previous preferences.

   Based on the availability and significance of long- and short-term interactions, the following context-adaptation scenarios are distinguished in [23]:

   a) *Last-N interactions:* Only the most recent N user actions are taken into account in the last-N interactions-based recommendation.
   b) *Session-based recommendations:* are made using only the user's most recent sequence of actions, which are only known for the period of a session (the particular duration during which the user interacted with the website).
   c) *Session-aware recommendations:* are made when it is known both what users did in the most recent session and what they have done in the past. If there are identifiable returning consumers, this type of issue will arise.

2) Trend detection: Another feasible, if less researched, objective that can be achieved is trend detection. Trend detection is the identification of trends in a given sequence dataset. In [23]

the sequential log data can be divided into the following categories for the information that can be extracted and used in the recommendation process:

a) *Community trends:* SARS can try to find and use popularity patterns in the interaction logs to enhance the recommendations since the popularity of items may change over time in different domains.
b) *Individual trends:* Shifts in people's preferences for particular things are also possible. When there is a natural interest drift, certain changes in interest may result. Modeling the dynamics of user musical preferences is one such issue.

3) Repeated Recommendation: Recommending products that the user is already familiar with or has previously purchased can be effective in specific application domains. In the conventional matrix completion configuration, these eventualities are not at all taken into account. In [23] there are identified the following categories of scenarios:
   a) *Identifying patterns of frequently occurring user activity*: SARS can use historical interaction logs to discover patterns of frequently occurring user behavior. For example, SARS may propose launching the email or calls app after opening the contacts app, i.e., used to provide shortcuts.
   b) *Recurring recommendations as reminders*: Recurring recommendations can assist in reminding users of past interests. These reminders may be for items the user may have recently interacted with or may be for items they may have forgotten, depending on the domain, such as Amazon.

1.8 Session-Based Recommendation Systems

As seen in Section 1.7, session-based recommendations are considered one of the main computational task situations in sequence-aware recommender systems. An SBRS is a type of recommendation system that focuses on making personalized recommendations based on a user's current session or browsing behavior. Unlike traditional recommender systems that consider long-term user preferences and historical data, session-based recommendation systems prioritize the immediate context and the user's short-term interests.

In a session-based recommendation system, the user's current session is treated as a sequence of interactions or actions, such as clicks, page views, or purchases, within a specific time frame. These interactions capture the user's preferences and interests in the present moment. The system analyzes this sequential data to generate real-time recommendations that cater to the user's immediate needs. Session-based recommender systems aim to predict the unknown part of a session (see Figure 6) or the future sessions based on modeling the complex relations embedded within a session or between sessions [24].Table 1 presents a comprehensive comparison between SBRS and other typical RSs.

Figure 6 : An example of an e-commerce website using SBRS

	SBRS	***Collaborative Filtering (CF)***	***Content-Based Filtering (CBF)***	***Hybrid Recommender Systems***
Data Source	User session interactions	User-item interactions	Item features	Multiple sources (session, CF, CBF)
Recommendation Type	Session-based recommendations	User-based or item-based recommendations	Item-based recommendations	Combined recommendations
Personalization	Highly personalized	Moderately personalized	Moderately personalized	Highly personalized
User Context	Captures temporal context	Temporal context limited	No explicit context captured	Captures multiple types of context
Sequential Patterns	Exploits sequential behavior	Doesn't consider sequential patterns	Doesn't consider sequential patterns	May consider sequential patterns
Cold Start Problem	Suffers from cold start problem	Suffers from cold start problem	Less affected by cold start problem	Moderate impact
Scalability	Can face scalability challenges	Scalable	Scalable	Scalable
Sparsity Handling	Handles sparse session data effectively	Requires denser data for accuracy	Handles sparse data effectively	Handles sparse data effectively
Serendipity	Can offer serendipitous recommendations	Moderate serendipity	Limited serendipity	Moderate serendipity
Explanation	Limited explicit explanations	Lacks explicit explanations	May provide feature-based explanations	May provide combined explanations

Table 1: Comparison between SBRS and other RSs

1.8.1 Categorization

An SBRS makes the unknown session information as its target to be predicted by taking the prior session information as the context and condition, which is called a session context C in this work. A session context can either be an intra-session context C^{la} or an inter-session context C^{le} according to whether the session context comes only from one session (the current one) or across multiple sessions (before the current one) [19].

According to [24], SBRS can be generally categorized into two major branches:

- Next-item(s) Recommendations: given an intra-session context C^{la} over the current session S_n, the next-item(s) recommendations predict the next item(s) i_t in $S_n$conditional on C^{la}. Next-item(s) recommendations are the mainstream and the most common setting of session-based recommendations.
- Next-session (next-basket) Recommendations*:* given an inter-session context C^{le} for current session S_n, the next-session recommendations predict those items possibly occurring in session S_n.

1.8.2 Approaches

We try to classify SBRS in this section from a technical point of view all currently published efforts are specifically divided into two branches: model-free techniques and model-based approaches, various types of techniques are present in each branch (Figure 7).

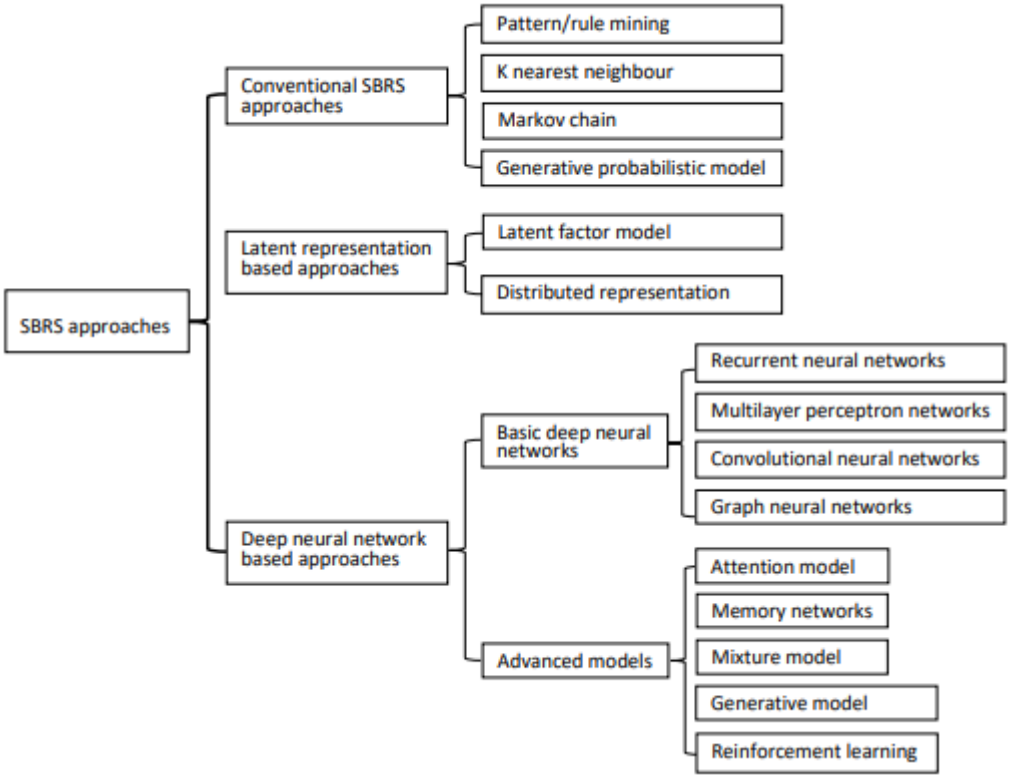


Figure 7 : The classification of SBRS's approaches

1.8.2.1 Model-Free Approaches

Model-free approaches are typically devoid of complicated mathematical models and are based primarily on data mining techniques. Sequential pattern-based RS for ordinal session data and pattern/rule-based RS for unordered session data are two common techniques in this area.

a) *Pattern/Rule-based Approaches:* Pattern/rule-based RS first mine frequent patterns or association rules and then use these patterns and rules to guide the subsequent recommendations. This is based on the assumption that most customers would follow the common shopping patterns [25].For instance, customers usually bought milk and bread

together when they go shopping, therefore {milk, bread} can be treated as a frequent pattern to recommend bread to those why have bought milk. It should be noted that pattern/rule-based RS are applied in unordered data [26].

b) *Sequential Pattern-based Approaches: To* handle those data having a strict order over items or involving time-factor based effect, sequential pattern-based RS are proposed. Similar to pattern-based RS, they first mine a collection of sequential patterns and then recommend the remaining items after the occurrence of the prior items [27].

1.8.2.2 Model-based Approaches

Model-based RS are typically constructed on stringent assumptions like ordering over objects and intricate models like Markov chain models, in contrast to model-free RS. The three primary groups of model-based approaches that are now in use are Markov Chain-based approaches, factorization-based approaches, and neural model-based approaches.

a) Markov Chain-based Approaches

Using transitional probabilities, Markov Chain-based RS models the first-order (and sometimes higher order) dependency over a sequence of items, and then uses that dependency to create recommendations for the next items. Markov Chain-based RS take into account all things and so significantly reduce information loss, in contrast to sequential pattern-based techniques that are simple to filter out such infrequent items and patterns and consequently result in information loss [28].

b) Factorization-based Approaches

These methods create a latent representation vector for each item by first factoring the item co-occurrence matrix or the item-to-item transitional matrix [29]. Then they use these latent representations to predict the subsequent items. Such methods should be distinguished from the frequently employed factorization machine (similar to matrix factorization) in collaborative filtering-based RS, which typically factors the user-item interaction matrix (similar to rating matrix) into latent factors of users and items [15].

c) Neural Model-based Approaches

take advantage of the neural network to learn the complex relationships and interactions over items within or between sessions and then generate recommendations based on such interactions. Based on the model structure, neural model-based approaches can be divided into shallow neural model-based approaches, which sometimes are also called embedding models or representation learning models and deep neural model-based approaches like RNN [30].

Approaches	*Working mechanism*	*Applicable scenarios*	*Target issues*	*Pros*	*Cons*
Pattern/rule-based approaches	Mine frequent patterns or association rules to guide recommendations	Simple, balanced and dense session data without order	Capture explicit co-occurrence-based dependency between items	Intuitive, simple and effective on simple data	Information loss, cannot handle complex data (e.g., imbalanced data, long tailed data)
Sequential pattern-based approaches	Mine sequential patterns to guide recommendations	Simple, balanced and dense session data with order	Capture explicit co-occurrence-based inter-item sequential dependency	Intuitive, simple and effective to capture sequential relations on simple data	Information loss, cannot handle complex data (e.g., imbalanced data, long tailed data)
Markov Chain-based approaches	Use Markov chain to model the transitions between items or sessions for recommendations	Relative simple sequential data mainly with short-term and low-order dependency	Capture explicit or implicit inter-item sequential dependency	Reduced information loss, flexible, good at modelling short-term sequential dependency	Usually ignore long-term and higher-order dependency
Factorization-based approaches	Factorize item transitions into latent representations of items for recommendations	Relative simple data mainly with short-term and low-order dependency	Learn latent item representations to fit for item-to-item transitions	Reduced information loss, good at modelling low-order dependency	Easy to suffer data sparsity issues, cannot capture higher-order dependency
Neural model-based approaches	Model the complex dependency in a neural network and embed this dependency into the learned latent representations for recommendations	Complex data with sequential or non-sequential dependency	Encode the complex dependency into latent representations of items or sessions	Powerful, can capture both long term and short term, higher-order and low-order dependency	Hard to implement, computationally costly

Table 2 : Comparisons between Different Technical Approaches for Session-based Recommendations

1.8.3 Challenges and Limitations of SBRS

Session-based recommendation systems face several challenges, including [31]:

1. Data Sparsity: Because sessions are frequently short and only contain a limited amount of user choices, session data may be naturally sparse. This makes it difficult to accurately determine the user's interests and offer pertinent recommendations.
2. Dynamic User Preferences: During a single session, a user's preferences and intentions may quickly change. It might be challenging to accurately model and accommodate their dynamic choices because their demands and interests may change depending on the immediate situations.
3. Cold-Start Issue: When a new user begins a session, little is known about their preferences and tendencies. Without historical data, it is difficult to give individualized recommendations due to the cold-start problem.
4. Contextual Understanding: It might be challenging to comprehend a user's preferences and intents when sessions lack explicit user feedback or explicit item evaluations. Effective interpretation of the implicit signals from the session context is required by the recommendation system.
5. Focus on the Short Term: Session-based recommendation systems are primarily concerned with the user's plans for the immediate future, which may not fully reflect the user's long-term goals. This may result in recommendations lacking diversity or failing to take into account a user's broader preferences.
6. Session Ambiguity: Sessions can be ambiguous, and the system needs to disambiguate the user's intent from the observed behavior. Different users may have different motivations or goals for similar session behavior, making it challenging to generalize recommendations

1.9 Conclusion

In the beginning of this chapter, a survey of recommender systems, a description of the session-based recommendation problem, and a discussion of where it fits in the family of sequence-aware recommendations were provided. It generated a thorough comparison between SBRSs and other RSs, followed by the formations and associated definitions of SBRS, which in turn stimulated the need for session-based recommender systems research. Finally, approaches, challenges and difficulties were examined. The next chapter will introduce Graph ATtention networks as a recent approach for developing SBRS.

Chapter2. Graph ATtention Networks

Chapter 2 : Graph ATtention Networks

2.1 Introduction

Graphs are a common method of presenting structured data in many areas, such as social networks, natural language processing, and recommendation systems.

Therefore, in the field of machine learning, it is becoming more and more crucial to be able to evaluate and model graph-structured data. As a new method of modeling graph-structured data, Graph Attention Networks (GATs) were introduced in 2018 and are a relatively recent advancement in the field of graph neural networks.

This chapter provides an introduction to the core ideas required to comprehend Graph Attention Networks.

2.2 Artificial Intelligence

Artificial Intelligence (AI) refers to the effort to automate intellectual tasks normally performed by humans. It is a general field that encompasses various approaches, including machine learning and deep learning, that involve training algorithms to learn from data without being explicitly programmed. While the idea of AI has been around since the 1950s, it wasn't until the advent of machine learning that significant progress was made in this field. Machine learning is a subfield of AI that focuses on the development of algorithms that can learn from and make predictions on data. Deep learning, a subset of machine learning, is a more advanced approach that leverages deep neural networks to learn complex representations of data.

Early AI systems, such as chess programs, were based on hardcoded rules crafted by programmers, a paradigm known as symbolic AI. However, this approach proved to be intractable for solving complex, fuzzy problems such as image classification, speech recognition, and language translation. This led to the rise of machine learning as a new approach to AI [32].

2.3 Machine learning

The development of algorithms and statistical models that allow computer systems to automatically improve their performance on a given job constitutes the field of machine learning, which is a subset of artificial intelligence. To create predictions or choices, these algorithms use data to learn from, spotting patterns and relationships within the data.

Predictive analytics, computer vision, natural language processing, and speech recognition are just a few of the domains where machine learning is used. Machine learning is becoming an increasingly essential topic for businesses and organizations wanting to extract insights and gain a competitive edge as the amount of data created continues to expand exponentially [33].

2.3.1 Data representations

Data representation in machine learning refers to the process of transforming raw data into a format suitable for analysis and model training. Effective data representation is crucial because it determines the quality of features that can be extracted and the performance of machine learning models [34].

Here are a few typical representations of data for machine learning [35]:

- Numeric Representation: This is the most common data representation, using numeric values to represent data. Numerical data can be continuous (e.g., temperature, time) or discrete (e.g., age, counts). Numerical data is easy to process and can be used directly by many machine learning algorithms.
- Categorical Representation: Categorical data represents non-numeric values that belong to a particular class or categories. Examples of this are gender, color, or product type. Before passing categorical data to a machine learning model, techniques such as one-hot encoding or sequential encoding are often used to convert categorical data into numerical form.
- Text representation: Text data such as documents, tweets, or customer reviews require special display techniques. A common approach is to use vectorization methods such as bag-of-words or term-frequency-inverse document frequency (TF-IDF) to convert text to numeric feature vectors. Another approach is to use word embeddings such as Word2Vec or GloVe to capture semantic relationships between words.
- Image representation: Image data is often represented as a grid of pixels, with each pixel having a color intensity value. Convolutional neural networks (CNNs) are widely used to process and extract features from image data. Pretrained CNN models such as VGG, ResNet, or InceptionNet can be used to extract high-level features or as a basis for transfer learning.Time series representation: Time series data represents measurements or observations taken over time, such as stock prices, sensor readings, or weather data. Time series data is typically represented as a sequence of data points with timestamps. Techniques like sliding windows or Fourier transformations can be used to extract relevant features from time series data.
- Graph representation: Graph data represents relationships between entities using nodes and edges. Graphs can be used to model social networks, knowledge graphs, or biological networks. Graph Neural Networks (GNNs) were developed to process graph-structured data and allow feature extraction based on graph connections and topology.

2.3.2 Types of machine learning

Machine learning can be split into three major areas:

1. Supervised Learning: This learning process is based on the comparison of calculated and projected results, which means that learning entails calculating the error and altering the error to get the desired outcome.
   For instance, a data set of houses of a particular size with actual costs is offered. The controlled formula is then used to produce additional of these accurate results, such as what the price of a brand-new home would definitely be [36].
   Regression and classification issues are subsets of supervised learning problems:
   - A classification problem occurs when the output variable is a category, such as "red" or "blue" or "disease" and "no disease."
   - A regression problem occurs when the output variable is a real value, such as "dollars" or "weight".

   Recommendation and time series prediction are two frequent sorts of challenges constructed on top of classification and regression.

   Some well-known supervised machine learning methods are:

- Linear regression is used to solve regression problems.
- Random forests are used to solve classification and regression issues.
- Support vector machines are used to solve classification difficulties.

2. Unsupervised Learning: This learning process is dependent on a comparison of the calculated result and also the anticipated outcome, which means learning explains identifying the error as well as converting the error to achieve the anticipated result.
   Unsupervised learning differs from supervised learning in that there are no correct answers and no educators. Algorithms are left to their own devices to uncover and provide fascinating data structures.
   For instance, a specific collection of houses of a certain size with actual costs is provided following that, the controlled formula is to create more of these appropriate comments, such as
   what would undoubtedly be the cost of a new house [36].
   Clustering and association problems are two types of unsupervised learning challenges:

   - *Clustering:* A clustering problem is one in which you wish to uncover the underlying groups in data, such as categorizing consumers based on their purchase activity.
   - *Association rule learning:* the issue is one in which you wish to identify rules that characterize substantial chunks of your data, such as persons who purchase X also tend to buy Y.

   Unsupervised learning methods that are widely used include k-means for clustering issues and the Apriori algorithm for learning association rules.

3. Reinforcement Learning : Reinforcement learning is also distinct from unsupervised learning, which is often concerned with discovering structure buried in collections of unlabeled data [37].
   Reinforcement learning is predicated on results and how a representative should respond to them in a given environment to maximize some notion of long-term motivation. For a successful conclusion, a benefit is provided, while a fee is applied for an unsuccessful outcome. Because appropriate input/output sets are never presented, nor are sub-optimal activities clearly addressed, reinforcement learning differs from the supervised learning problem [36].

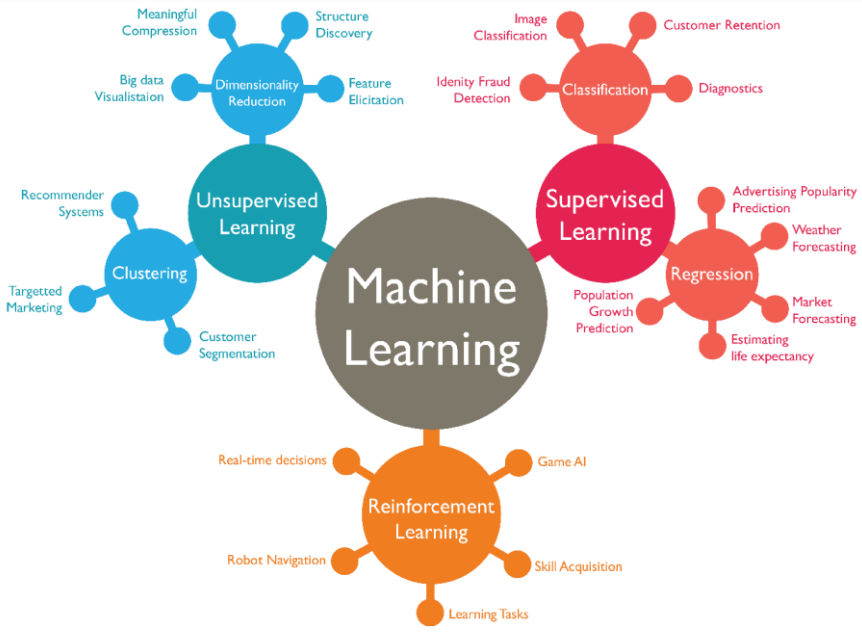


Figure 8 : Types of Machine Learning

2.4 Deep Learning and Neural Networks

Deep learning is a branch of machine learning that focuses on training artificial neural networks to learn and make choices in the same way that the human brain does. The term comes from the design of these neural networks, which are made up of multiple layers of linked nodes known as neurons. Deep learning algorithms may develop hierarchical representations of the input by processing data via various layers, allowing them to extract increasingly abstract features [38].

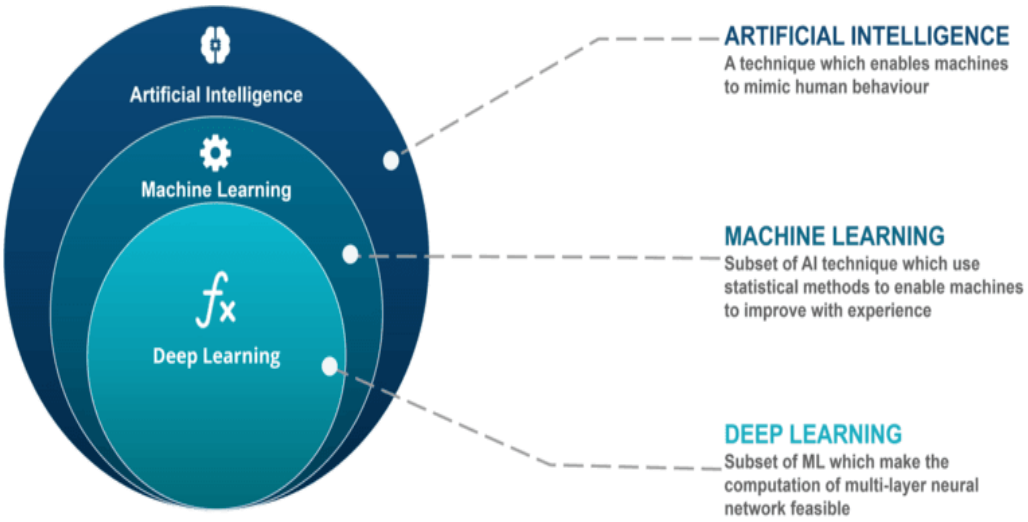


Figure 9: The relationship between AI, ML and DL

“Neural Networks is an algorithm that's get to know the hidden Link in a data set with a similar way as a human brain “

The subfield of machine learning called deep learning relies heavily on neural networks, which are among the most widely used and potent algorithms. Though neural networks may appear to be a mystery at first glance, with data flowing from the input layer into the "hidden layers," and then being processed in some unknowable way before emerging from the output layer, understanding the role of

these hidden layers are the crucial factor that determines the success of neural network implementation and optimization.

A neural network is defined as a computer system consisting of a series of simple but highly interconnected elements or nodes called "neurons", organized in layers, using dynamic responses to external information to process information.

This algorithm is useful for finding patterns too complex to manually extract and teach a machine to recognize. In this structure, the input layer introduces patterns into the neural network, and each component present in the input data has a neuron and communicates with one or more hidden layers in the network, which are simply called " hidden layers" .In these layers, all processing actually happens through a system Connections characterized by weights and biases.

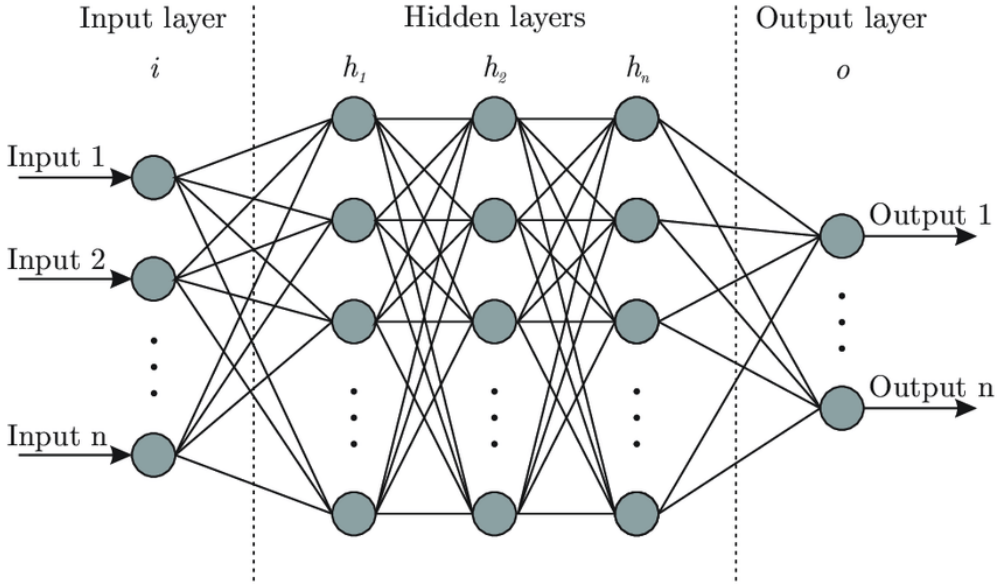


Figure 10: Artificial neural network architecture

2.4.1 Training Neural Networks

In a neural network, the fundamental building block of computation is the neuron, also referred to as a unit or a node. This component receives input from other nodes or an external source and performs calculations to produce an output. Every input has its own associated value. The value of an input is calculated through the assignment of a weight, denoted as 'w', which is determined in relation to the significance of other inputs. The node then applies a function to the sum of these inputs, which has been weighed accordingly [39].

At the start, a neural network's weights are assigned arbitrary values, which means the network is only capable of producing random alterations. It's naturally far from what it should be, resulting in a high loss score. However, with every round of processing an example, the optimizer adjusts the weights slightly in the right direction, which in turn lowers the loss score. This process, known as the training loop, is repeated multiple times (usually hundreds or even thousands of examples over tens of iterations) until the weight values are minimized and the loss function is reduced. A network that has undergone this process is referred to as a trained network, with minimal loss such that the outputs are as close as possible to the targets. This simple mechanism, once scaled, appears to be magical [40].

So, the process of training neural networks encompasses various stages that can be succinctly summarized as follows:

- *Data preparation :* Prepare training, validation, and test datasets. This includes cleaning the data, splitting it into training, validation, and test sets, and preprocessing the network input.
- *Network architecture design:* Choose the appropriate network architecture for the task at hand. This includes choosing the type of neural network (e.g., feedforward, convolutional, recurrent), the number of layers, and the number of neurons in each layer.
- *Initialization*: Initialize the weights and biases of the network. This can be done randomly or with pretrained weights.
- *Forward propagation:* Pass input data through the network to generate predictions.
- *Loss computation:* Computes the difference (i.e. loss or error) between the predicted output and the actual output.
- *Backpropagation*: Use backpropagation to compute loss gradients from network weights and biases.
- *Weight update*: The weights and biases of the network are updated to minimize loss using an optimization algorithm such as stochastic gradient descent.
- *Repeat*: Steps 4-7 are repeated for a fixed number of iterations (epochs) or until convergence.
- *Evaluation*: Evaluate the performance of the network against the test set.

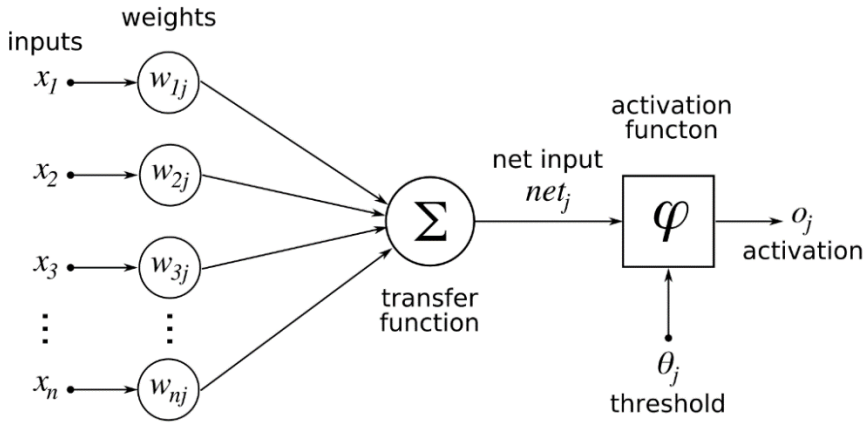


Figure 11: The learning process of deep learning

2.4.2 Back Propagation and Gradient Descent

Backpropagation is an important part of reducing error in a neural network model.

The backpropagation algorithm is frequently employed in machine learning. It operates by computing the gradient of the loss function, which serves as a guide towards the most optimal value for minimizing loss. To execute this, it utilizes the chain rule of calculus to calculate the gradient in reverse through the various layers of a neural network. Through the use of gradient descent, gradual progress towards the minimum value is made by taking small steps in the direction indicated by the gradient.

Gradient descent is the optimization algorithm used to update the weights of the network. It involves taking small steps in the direction of the negative gradient of the loss function. The learning rate determines the size of the steps [41].

Generally speaking, neural network or deep learning model training occurs in six stages:

1. Initialization: setting the corresponding activation a 1 for the input layer, and initializing weights w l for all layers.
2. Forward propagation[7] : For each l = 2, 3, . . ., L compute $Z^l = W^l a^{l-1} + b^1$and $a^l = f(Z^l)$.
3. Compute Error: defining an error function C, which captures the delta between the correct output and the actual output of the model, given the current model weights (in other words how far off is the model from the correct result).
4. Backpropagate The Error: Actual Output – Desired Output
5. Weight Update: changing weights and biases to the optimal values according to the results of the backpropagation algorithm
6. Iterate Until Convergence: Similarly, the network needs to iterate several times to learn. After each iteration, the backpropagation updates the weights towards less and less global loss function. At the end of this process, the model is ready to make predictions for unknown input data. New data can be fed to the model, a forward pass is performed, and the model generates its prediction.

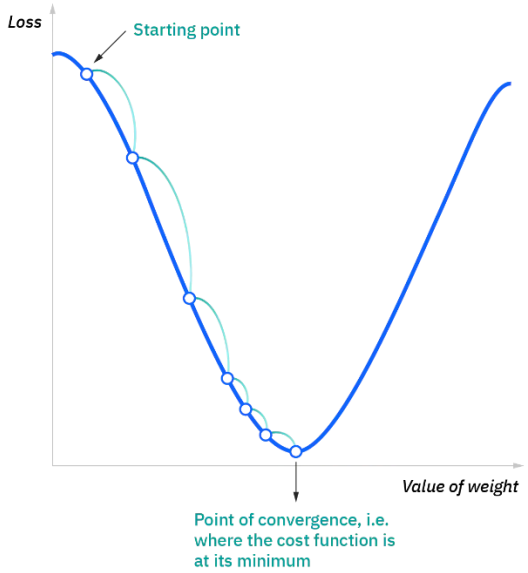


Figure 12: Gradient descent

2.4.3 Activation Function

An activation function is a mathematical equation that controls whether a neuron should be activated or not in a neural network model. It also helps to normalize the output of any input. There are a variety of activation functions that can influence the speed and how well a neural network converges, as well as sometimes prevent it from doing so [42].

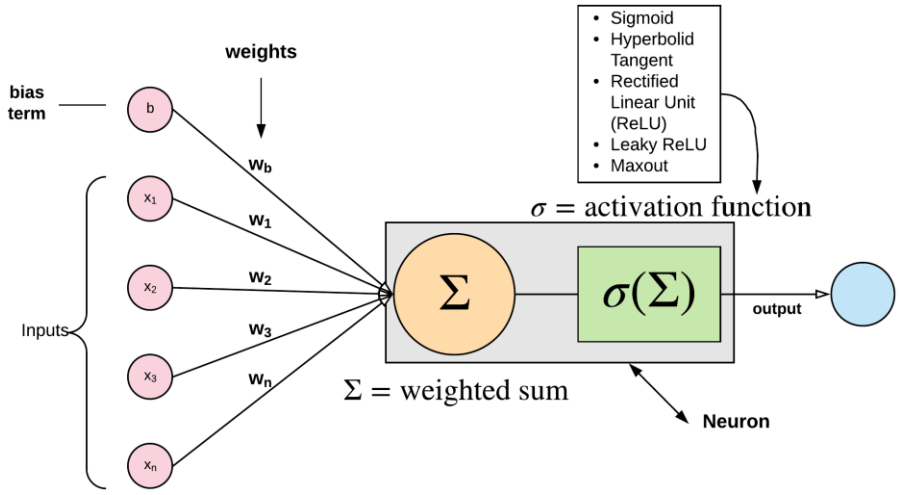


Figure 13: Activation function use in neural networks

1. Binary Step Function
   A step function with two possible outcomes is commonly known as a binary step function. The activation of a neuron in a binary step function is determined by a threshold value, which determines whether or not the neuron should be activated. When the input is supplied to the activation function, it is compared to a specific threshold. If the input is higher than the threshold, the neuron is stimulated, but if it is lower, it is inactive and its output is not transmitted to the succeeding hidden layer [43].
   Mathematically it can be represented as:

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x > 0 \end{cases}$$

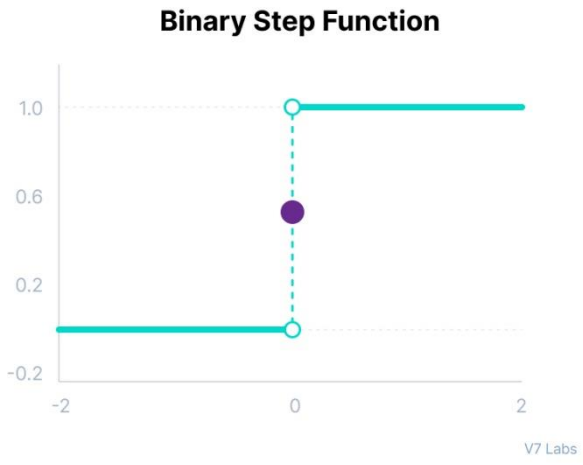


Figure 14: Binary step function

2. Linear Activation Function
   The activation function that follows a linear relationship between input and output is known as the Linear Activation Function. The identity function, or "no activation," is a linear activation function where the input and activation are directly proportional to each other, with the identity function being multiplied by x1.0. The function's output only reflects the value it receives and does not affect the weighted sum of the input in any way [43].

Mathematically it can be represented as:

$$f(x) = x$$

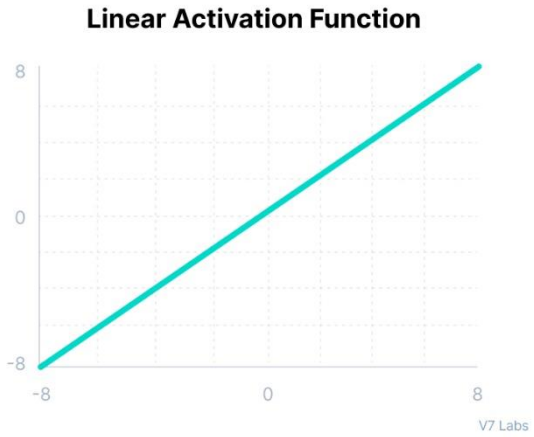


Figure 15: Linear activation function

3. Non-Linear Activation Functions

The activation function that is displayed linearly above is nothing more than a model of linear regression. Due to its limited capability, the model is unable to establish intricate connections between the inputs and outputs of the network [43].

The limitations of linear activation functions are resolved by non-linear activation functions, which offer the following solutions [43]:

- Backpropagation is permitted due to the relationship between the derivative function and the input, which enables a retrospective analysis of the weights in the input neurons that could potentially enhance the accuracy of predictions.
- Neural networks allow for the stacking of several layers of neurons, resulting in a non-linear combination of input that has passed through multiple layers. Any output can be expressed as a functional computation within a neural network.

Here some common non-linear activation functions are:

- *Sigmoid / Logistic:* sigmoids can reduce extreme values or outliers in data without removing them:

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

The reason why the sigmoid/logistic activation function is extensively used is due to the following factors:

1. When creating models that require prediction of probabilities, it is typical to use the sigmoid function as the ideal choice. The reasoning behind this is that probabilities can only exist within the range of 0 and 1, and sigmoid is able to accommodate this range.

2. The sigmoid activation function is identifiable by its distinctive S-shape, which represents the function's differentiability and capacity to provide a fluid gradient. This allows for a lack of abrupt changes in the output values of the function.

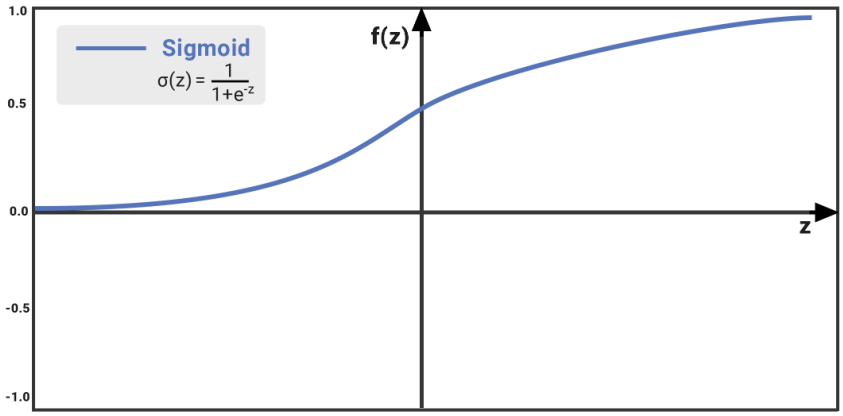


Figure 16: Sigmoid activation function graph

- *TanH / Hyperbolic Tangent :* The Tanh function is known for its resemblance to the sigmoid or logistic activation function, as both share the same S-shaped curve. However, the Tanh function has a distinct output range of -1 to 1. As the input value grows larger and more positive, the output value will approach 1.0. Conversely, as the input value becomes smaller and more negative, the output will approach -1.0.

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

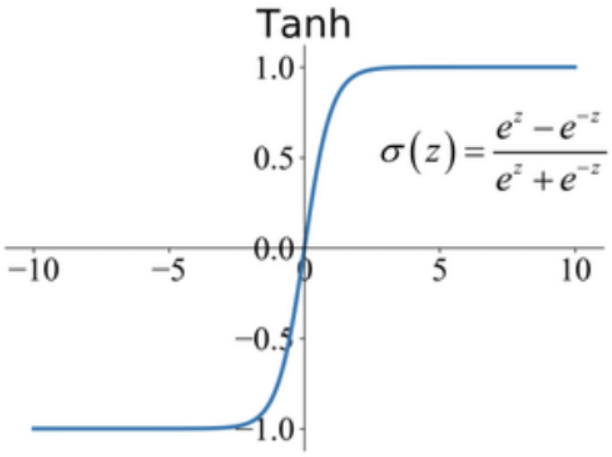


Figure 17: TanH activation function graph

- *ReLU (Rectifed Linear Unit) :* The rectified linear activation function or ReLU is a non-linear function or piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It is the most commonly used activation function in neural networks, especially in Convolutional Neural Networks (CNNs) & Multilayer perceptrons [44].

  ReLu is a non-linear activation function that is used in multi-layer neural networks or deep neural networks. This function can be represented as:

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

where x an input value. According to equation , the output of ReLu is the maximum value between zero and the input value.

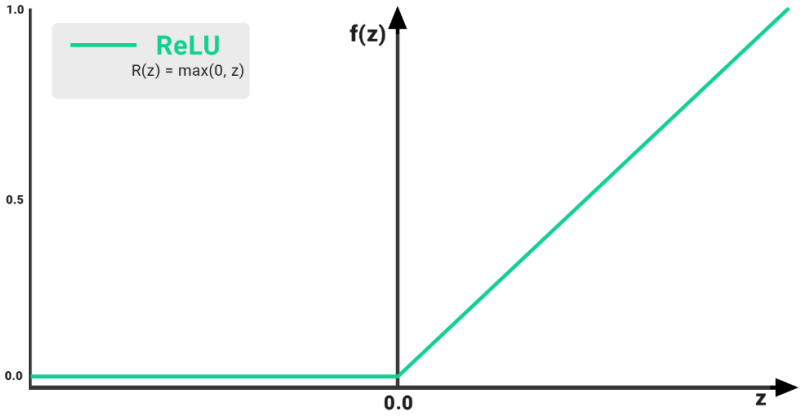


Figure 18: ReLU activation function graph

- *Leaky ReLU* : is an activation function used in artificial neural networks. It is similar to the standard ReLU function, except that it has a small non-zero output for negative input values. This allows the network to learn faster and reduces the risk of the network getting stuck in a local minimum. Leaky ReLU is popular in tasks where we may suffer from sparse gradients, for example training generative adversarial networks [45].

$$f(x) = max(0.1x, x)$$

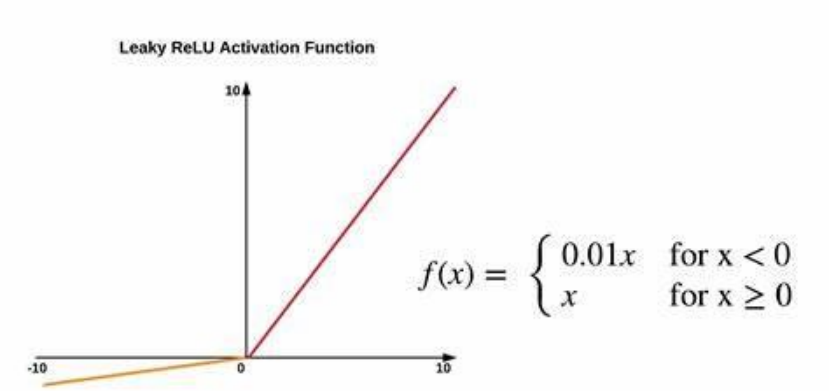


Figure 19: Leaky ReLU activation function graph

- *Softmax :* In machine learning, the Softmax function is frequently used. It is often used to convert a set of values into a probability distribution. This enables us to forecast the likelihood of a specific result. The function is computed by taking the exponentiation of each array member and dividing it by the total of all exponentiated array values. As a consequence, a probability distribution is represented by a set of values that add to one.

$$softmax(Z_i) = \frac{exp(Z_i)}{\sum_j exp(Z_i)}$$

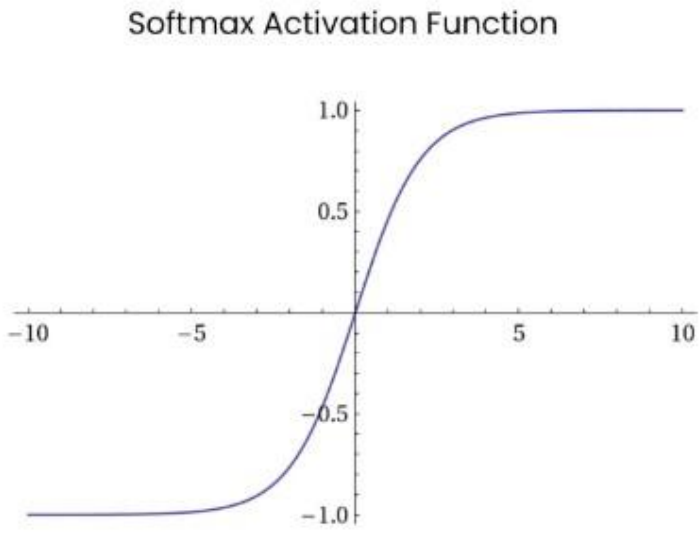


Figure 20: Softmax activation function graph

2.4.4 Loss Function

A loss function is a critical tool in evaluating the effectiveness of a neural network for a specific task. The method to accomplish this is relatively straightforward: for each training example, the network processes the data to obtain a numerical value. Then, this value is subtracted from the desired result, and the difference is squared. This is done because both positive and negative differences are equally undesirable.

$$L(y, \hat{y}) = \frac{1}{m}\sum_{i=1}^{m}(y_i, \hat{y}_i)^2$$

In our neural network, y is the desired output we aim to obtain, while y with a hat represents the actual output we receive after processing a given example through the network. The index of a training example is denoted by i. For instance, let's take the example of dogs-vs-cats dataset, which contains images of labeled dogs and cats, with one representing a dog and zero representing a cat. The label we want to obtain from the network after passing the image through it is represented by y. To calculate the loss function, we must iterate over each training example in the dataset, calculate y for each example, and then apply the function as per the formula defined above. A higher value of the loss function indicates poor performance by the network; hence, we aim for the smallest possible value. We can further comprehend the relationship between the loss function and the neural network by substituting y with the network's actual output in the formula.

2.4.4.1 Loss Functions for Regression

- Mean squared error loss (MSE): The squared loss function is utilized when working on a regression model that requires a real-valued output.
  Consider the scenario when just one output feature must be predicted; the error in a forecast is squared and averaged over the number of data points, basic and straightforward [46].

$$L(W, b) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

- Mean absolute error loss (MAE): In the same vein, (MAE) is similar to (MSE) showing in the following equation: [45]

$$L(W,b) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\widehat{y_{ij}} - y_{ij}\right|$$

- Mean Squared Log Error (MSLE): another function used for regression with this equation [46]:

$$L(W,b) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\log\widehat{y_{ij}} - \log y_{ij}\right|$$

2.4.4.2 Loss Functions for Classification

When developing neural networks for classification issues, the emphasis is frequently placed on assigning probabilities to these classifications [46].

These differing conditions necessitate distinct loss functions:

- Hinge loss

When the network must be tuned for a hard classification, the most commonly utilized loss function is the hinge loss and it is mostly used for binary classifications. There are extensions for multiclass classification [46].

$$L(W,b) = \frac{1}{N}\sum_{i=1}^{N} max(0,1 - \hat{y}_i * y_i)^2$$

- Logistic loss

they are utilized When probabilities are more interesting than hard classifications [46].

$$L(W,b) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\widehat{y_{ij}} - y_{ij}$$

2.4.4.3 Loss Functions for Reconstruction

This set of loss functions is related to the concept of reconstruction. The concept is straightforward. A neural network is taught to precisely replicate its input [46].

2.5 Deep Learning challenges

Deep learning involves training complex neural networks with a large number of parameters, which can lead to a number of challenges related to testing, validating, and overfitting.

2.5.1 Testing, Validating and Overfitting

Once a model has been trained, it is crucial to ensure that it can effectively generalize to new cases, rather than relying on mere hope. To achieve this, the model needs to undergo evaluation and fine-tuning if necessary. The most reliable way to assess the model's generalization capabilities is by testing it on new cases. One approach involves deploying the model into production and monitoring

its performance. Although this method can be effective, if the model proves to be flawed, it may result in user complaints, making it less than ideal. A better alternative is to divide the available data into two distinct sets: the training set and the test set. As their names suggest, the model is trained using the training set, while the test set is employed to evaluate its performance.

The measure of the model's error on new cases is known as the generalization error or out-of-sample error. By evaluating the model on the test set, one can obtain an estimation of this error. This value provides valuable insights into the model's performance on instances it has not encountered during training. In cases where the training error is low, indicating that the model makes few mistakes on the training set, but the generalization error is high, it signifies that the model is overfitting the training data. Preventing poor generalization necessitates the ability to halt the training process as soon as overfitting commences.

To address this concern, the training process is divided into epochs. An epoch refers to a single iteration over the entire training set. Specifically, for a training set of size 'd' and performing gradient descent with a batch size of 'b', an epoch would encompass 'd/b' model updates. At the conclusion of each epoch, it becomes essential to measure how effectively the model generalizes. To accomplish this, an additional validation set is utilized. The validation set provides insights into the model's performance on data it has not encountered previously. If the accuracy on the training set continues to improve while the accuracy on the validation set remains stagnant or deteriorates, it serves as an indication that training should be stopped to avoid overfitting [47] [48] .

It is worth noting that the opposite of overfitting, known as underfitting, also exists. Underfitting occurs when the model is too simplistic to grasp the underlying structure of the data. For instance, a linear model attempting to predict life satisfaction is susceptible to underfitting, as reality is more intricate than the model's capabilities, leading to inaccurate predictions even on the training examples.

2.5.2 Hyperparameters

In machine learning, hyperparameters are model parameters that cannot be learnt from training data and must be defined before training begins. They have control over parts of the training process such as the model's learning rate and the number of iterations through the training data. Learning rate, regularization strength, the number of hidden layers in neural networks, and the number of trees in a random forest model are all examples of hyperparameters. The selection of hyperparameters may have a substantial impact on the model's performance and generalization ability, therefore determining the best hyperparameters is an important stage in the machine learning workflow [49].

2.6 Graph Neural Networks

A graph neural network (GNN) is a neural network that uses a graph data structure that encodes relationships between entities. GNNs are used to learn representations of nodes and edges on graphs and can be used for various tasks such as node classification, link prediction, and graph classification.

GNNs typically involve the propagation of node features in a graph, using message passing techniques to update features based on those of neighboring nodes. The architecture of GNNs can vary depending on the specific task and the type of graph to be analyzed. There is different types of GNNs, ranging from traditional models such as Graph Convolutional Networks (GCN) to recent advances such as Graph Attention Networks (GAT) and GraphSAGE [37].

A graph neural network (GNN) has two distinguishing attributes:

- It takes a graph as input.

- It produces permutation equivariant results.

GNNs are designed to build node representations that take into account both the topology of the graph and any accessible feature information.

2.6.1 Main Concepts

In the following section we are going to describe the main concepts that enable graph neural networks: the nature of non-Euclidean data, graph neighborhoods, concepts of neural message passing.

2.6.1.1 Non-Euclidean space data

The majority of deep learning techniques that exist today are designed to work with structures that are either Euclidean or grid-like in nature, such as images, videos, or textual data. For instance, images can be thought of as a function on the Euclidean space (plane), sampled on a grid, which allows us to leverage their local connectivity and use Convolutional Neural Networks that utilize this information about the data. Similarly, textual data can be represented as a sequence on a Euclidean plane, which also has structural concepts of "before" and "after" that NLP models take advantage of. However, there are many other types of data - such as social networks in computational social sciences, sensor networks in communications, molecule structures in computational chemistry, and meshed surfaces in computer graphics - that do not fit the Euclidean mold, and can instead be categorized as non-Euclidean space data. The non-Euclidean nature here generally means that there are no common systems of coordinates, data priors or common structures that represent such data [50]. Therefore, basic approaches that work on Euclidean data, fail to work on its generalization, non-Euclidean case, where prior structure can be arbitrarily represented. It is also worth noting how the Euclidean data can be seen as a particular case of non-Euclidean data. For example, an image grid of N × N pixels can be viewed as a graph with N2 nodes and at most 8 edges per node (connecting to the nearest grid of pixels) with each node associated a feature vector representing the image’s pixel intensity.

2.6.1.2 Graph neighborhood

In graph theory, the neighborhood of a node inside a graph refers to the collection of nodes that are immediately related to it by an edge. In simple terms, it includes the nodes that are close to the given node.
In an undirected graph, the neighborhood of a node v is defined mathematically. G is officially defined as $N(v) = \{u \in V(G) : uv \in E(G)\}$, where V (G) is the set of vertices in G and E(G) is the set of edges in G.

Understanding a node's neighborhood is critical in graph theory because it gives insights into the local structure of the graph surrounding that node. For example, the size of a node's neighborhood might indicate its degree of centrality.

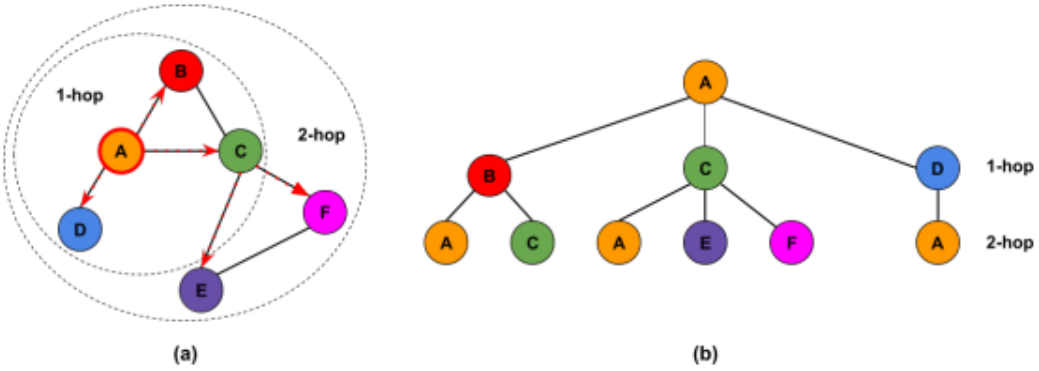


Figure 21: 1-hop and 2-hop neighborhoods of a given target node A.

2.6.1.3 Permutation equivariance and invariance

In GNNs, *permutation equivariance* indicates that the network's output stays constant when the nodes in the input graph are permuted. If we shuffle the node order, the GNN should provide the same output, but with the node order reflecting the new permutation. This characteristic enables the GNN to identify the graph's underlying symmetry and deliver symmetric results [51].

Permutation invariance Operations applied to graph data must be permutation-invariant, i.e. independent of the order of neighbor nodes, as there is no specific way to order them [51].

Permutation equivariance and permutation invariance are critical properties of Graph Neural Networks (GNNs). These properties are especially important in situations when node ordering does not transmit meaningful information or when the network structure stays unaltered after permuting its nodes. GNNs with these features can successfully capture the local structure and linkages existing in the network without being influenced by node order. This capacity to generalize and generate reliable representations is critical in tasks like node classification, graph classification, and link prediction [51].

2.6.1.4 Neural message passing

The term "neural message passing" refers to a specific framework utilized within the context of graph neural networks (GNNs). This framework enables the exchange and assimilation of information between the nodes present within a graph.

At its core, neural message passing assumes that each node u ∈ V in a given graph G = (V, E), is associated a feature vector xu ∈ R d, where d is some feature dimension. In order to update a node u's feature vector, producing a new feature vector h (k) u , we need to be able to collect feature information coming from the node's neighbors, as well as to integrate this information, to produce a new representation for the node This procedure is reiterated for several message passing steps, enabling the nodes to exchange information and enhance their representations.

The equation for this process can be articulated in the following manner:

$$2.\, h(k+1)_u = UPDATE(k)\big(h(k)_u, AGGREGATE(k)(\{h(k)_v, \forall v \in N(u)\})\big)$$

In this context, the functions UPDATE and AGGREGATE are differentiated by their arbitrary characteristics, which are usually established as neural networks. The AGGREGATE function collects the embeddings of surrounding nodes in the neighborhood N(u) to produce a message

m(k)_N(u) based on this compiled data. Conversely, the UPDATE function combines the message m(k)_N(u) with the previous embedding h(k-1) u to create the new and improved embedding h(k)_u. All nodes begin with initial embeddings set to the input features xu for all nodes u at iteration k=0. By repeating the process of message-passing K times, the desired outcome is achieved, we can use the final layer output to define the embeddings for each node [52] .

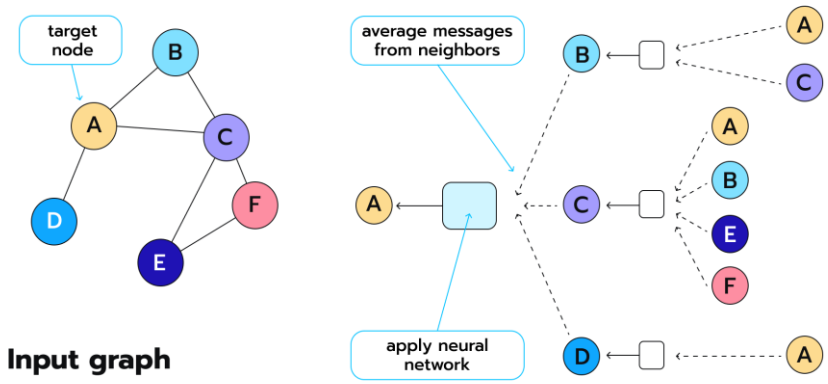


Figure 22: Neural message passing of target node A

2.6.2 Types of Graph Neural Networks

1. *Graph Convolutional Networks (GCNs):* A GCN is a type of GNN that uses convolutional layers to process graph data. These layers apply a set of learnable filters to the graph, designed to take into account the structure of the graph and the relationships between vertices.

   Some of the most common uses of GNNs in graph classification include:

   - Social Network Analysis: GNNs can classify people in a social network based on their relationship to other people.
   - Bioinformatics: GNNs can classify protein-protein interaction networks and predict protein functions.

2. *Graph Autoencoders (GAEs):* GAEs use graph convolutional layers to learn a low-dimensional representation of the input graph. The network is trained to encode the graph into a lower dimensional space, and decode it back to the original graph.

3. *Graph Recurrent Networks (GRNs):* GRNs are designed for processing graph-structured data in a sequence or time-series setting. They use recurrent neural networks to propagate information between nodes across several time-steps.

4. *Graph Transformers:* Graph Transformers are inspired by the Transformer architecture used for natural language processing tasks. They use self-attention mechanisms to capture the relationships between nodes in the graph.

5. *Graph Attention Networks (GATs):* GAT is a GNN that uses an attention mechanism to weigh the importance of different vertices in a graph when processing data. This allows GNNs to focus on the most relevant cornerstones and relations when making predictions.

2.7 Graph ATtention Networks

In a research paper titled "Graph Attention Networks" published in 2018, Petar Velikovi and colleagues introduced the Graph Attention Network (GAT), a form of Graph Neural Network (GNN).

GAT was created in order to overcome some of the limitations of conventional GNNs when modeling complicated dependencies and relationships in graph-structured data. Typical traditional GNNs treat each neighbor equally when aggregating input from nodes in the neighborhood. However, many nodes may have differing degrees of importance or relevance in many real-world settings.

To tackle this challenge, GAT incorporates attention mechanisms into the GNN architecture. Attention mechanisms allow the network to dynamically weigh the importance of different nodes when aggregating information. In other words, GAT assigns attention coefficients to each pair of nodes in the graph, indicating the importance of one node's information for another. These coefficients are learned during the training process.

By leveraging attention, GAT enables nodes to focus on the most relevant neighbors and weigh their contributions accordingly. This adaptive aggregation process allows GAT to capture more nuanced relationships and dependencies within the graph, leading to improved performance in various graph-related tasks [53].

2.7.1 Architecture of Graph ATtention Networks

The architecture of a Graph Attention Network (GAT) typically consists of the following components [53]:

1. Input Graph: GAT takes as input a graph represented by nodes and edges. Each node is associated with a feature vector, which could encode attributes or characteristics of the node.
2. Node Embedding Layer: The initial node features are fed into a node embedding layer, which maps each node's feature vector to a lower-dimensional space. This layer aims to capture meaningful representations of the nodes.
3. Graph Attention Layer(s): The core component of GAT is the graph attention layer. It consists of multiple attention heads, which operate in parallel to learn different sets of attention coefficients. Each attention head computes attention scores between a central node and its neighboring nodes.

   a. *Attention Coefficient Computation*: For each pair of nodes (a central node and one of its neighbors), attention coefficients are computed. The coefficients measure the importance of the neighbor's information for the central node and are calculated based on their respective feature vectors.

   b. Attention Aggregation: The attention coefficients are used to compute weighted aggregations of the neighboring node features. This aggregation step ensures that the central node can focus on the most relevant information from its neighbors.

   c. Multi-Head Attention: The outputs of the attention heads are concatenated or averaged to produce a combined representation for each node. This allows the model to capture different aspects or perspectives of the graph.

4. Optional Additional Layers: Depending on the specific task, additional layers such as fully connected layers or pooling operations can be added to further process the node representations and perform higher-level computations.
5. Output Layer: The final node representations obtained from the graph attention layers are fed into an output layer for the specific task at hand. This could be a node classification layer, link prediction layer, or any other layer appropriate for the target problem.

The parameters of the GAT model, including the attention coefficients, are learned through backpropagation and optimization techniques such as gradient descent, minimizing a task-specific loss function.

2.7.2 Graph ATtention Networks advantages

The Graph Attention Network (GAT) model offers several advantages, including:

1. Adaptive Information Aggregation: GAT involves an attention mechanism that supports adaptive information aggregation. It allows nodes to dynamically weigh the importance of neighboring nodes when aggregating information. This adaptability helps capture complex relationships and dependencies in the graph, thereby improving performance.
2. Flexible contextual representation: GAT captures contextual information by considering related nodes in the graph. By focusing on the most informative neighbors, GAT generates node representations responsive to local context. Because of this flexibility and context awareness, GAT is well-suited for tasks where local neighborhood plays a key role.
3. Multi-Head Attention Mechanism: GAT uses a multi-head attention mechanism, which uses multiple attention heads to simultaneously capture different aspects or views of a graph. This allows the model to learn diverse and complementary representations. The combination of multiple heads increases the expressiveness of GAT and improves the performance of various tasks.
4. Handling large and sparse graphs: GAT works well even when dealing with large and sparse graphs due to its ability to selectively consider relevant nodes. By focusing on the most important nodes and considering the interactions between them, GAT efficiently handles graph-structured data, making it applicable to real-world scenarios with complex and large-scale graph structures.
5. Versatility and generalizability: GAT have been successfully applied to various graph-related tasks, including node classification, link prediction, recommender systems, and graph-level prediction. Its flexibility and ability to capture complex dependencies make it a versatile model that can be adapted to different applications.

It's important to note that the advantages of GAT may depend on the specific problem, dataset, and implementation details. It's always recommended to carefully evaluate the model's performance and compare it with other baselines or state-of-the-art methods for a given task.

2.7.3 Comparison of GAT and different GNN's architectures

Model	***Key Idea***	***Attention***	***Aggregation***	***Message Passing***	***Advantages***
Graph Attention Network (GAT)	Attention-based message passing	Yes	Weighted sum	Linear combination of node features and attention scores	Captures node importance, allows for flexible feature weighting
Graph Convolutional Network (GCN)	Convolution-based message passing	No	Averaging	Convolution of node features and adjacency matrix	Simple, easy to implement
Graph Isomorphism Network (GIN)	Learnable function-based message passing	No	Multi-layer perceptron	Aggregation of node features and fixed function	Powerful, can learn complex functions
GraphSAGE	Sampling-based message passing	No	Max pooling or concatenation	Aggregation of node features from sampled neighborhood	Scalable to large graphs
GNN-LSTM	LSTM-based message passing	No	LSTM	LSTM-based message passing	Captures temporal dependencies

Table 3: Comparison of GAT and different GNN's architectures

2.8 Conclusion

In this chapter, we have defined the main concept of deep learning as a subset of the field of machine learning, which is a subfield of AI, and at the very core are graph neural networks (GNNs), The terms deep learning and neural networks in reality encompass a wide variety of architectures. Most of these networks will share elements i.e., gradient descent, the backpropagation algorithm activation functions, loss functions. While the space of models is diverse, most of them can be grouped into some broad categories. One of the major ones was the subject of the last section: Graph Attention Networks (GATs)

In summary, the subsequent chapter will delve into the practical use of GATs, providing a comprehensive understanding of their applications in machine learning. By applying the concepts and methodologies presented thus far.

Chapter3.

Graph ATtention Networks for the development of session Based recommendation systems

Chapter 3 : Graph ATtention Networks for the development of session-based recommendation systems

3.1 Introduction

E-commerce platforms frequently inundate their users with an overwhelming number of products for sale. To improve user the experience, recommender systems provide customized and practical recommendations. However, conventional recommender systems rely on user profiles, which can be difficult to create for new users, anonymous visitors, or individuals who have deleted their tracking data. In such circumstances, session-based recommendations provide an alternative method.
The following chapter delves into the practical application of Graph Attention Networks (GATs) in the context of session-based recommendations. The utilization of graph attention mechanisms allows us to enhance the precision and efficacy of customized recommendations, leading to a personalized and uninterrupted shopping experience for all users. GATs provide a powerful tool to analyze the browsing behavior within the current session and generate relevant suggestions based on this context. This chapter delves into the methodologies and techniques employed to optimize GATs for session-based recommendations, with the goal of delivering enhanced user satisfaction and engagement.

3.2 Deep learning-based recommendations

In just a few years, numerous deep recommender systems have been put out. Innovation is definitely thriving in the industry. It would be simple to dispute the necessity of using so many distinct architectures at this point and perhaps even the effectiveness of neural networks for the given problem domain. To this purpose, the benefits of deep learning-based recommendation models were outlined as follows in [54]:

- *Nonlinear Transformation*: contrary to linear models, deep neural networks is capable of modeling the non-linearity in data with nonlinear activation functions such as relu, sigmoid, tanh, etc. This property makes it possible to capture the complex and intricate user-item interaction patterns.
- *Representation learning*: Learning complex patterns and representations from large-scale and high-dimensional data is where deep learning models excel. They can automatically learn rich representations of items, users, and their interactions, capturing intricate relationships and latent factors that conventional methods may struggle to uncover.
- *Flexibility and scalability*: Deep learning models offer flexibility in modeling different types of recommendation tasks. They can handle various data types such as textual, visual, and sequential data, enabling the incorporation of diverse information sources into the recommendation process. Moreover, deep learning models can scale well to handle large datasets and are amenable to distributed computing frameworks.

3.3 A GAT Based-Model for Session Based Recommender Systems

A Graph Attention Network (GAT) can be used as a model for session-based recommender system to recommend items to users based on their current session or sequence of interactions. . By leveraging attention mechanisms and graph convolutions, Graph Attention Networks excel in

capturing the dependencies and interactions among items in a session-based recommendation system. They can effectively model the dynamic nature of user behavior and provide accurate recommendations based on the sequential patterns observed in the session data.

- **Architecture of the proposed model**

The proposed model based on GAT for building SBRS is depicted in Fig 23. After representing a session with a graphical structure (nodes and edges), this model processes the resulting graph with its adjacency matrix through several layers iteratively.

1. *Input layer:* The layer receives a set of node features as its input h= $\{\vec{h}_1, \vec{h}_2, \vec{h}_3, \ldots, \vec{h}_N\}$, $\vec{h}_i \in R^F$ where N is the number of nodes, and F is the number of features in each node The layer outputs a new set of node features of the form (of potentially different cardinality F'), h'= $\{\vec{h'}_1, \vec{h'}_2, \vec{h'}_3, \ldots, \vec{h'}_N\}$, as its output.
2. *Linear Transformation layer:* To represent each node in a lower dimension, the feature matrix x (set of $\vec{h}_i$) is transformed using a shared weight matrix W and a bias b to produce the output Y.

$$h^* = Wh + b$$

This linear transformation allows making the required features simpler to identify and classify to compute the attention coefficients on a reduced feature space.

3. *Attention mechanism Layer*: an attention mechanism is used to compute the importance of each neighboring node for a given node i. Two main operations are carried out in this layer, namely:
   a. Attention Coefficients: The attention coefficients are computed by a shared neural network with parameters of {a}. The output of this network is a scalar value $\boldsymbol{e_{ij}}$ that represents the compatibility between nodes i and j.

$$3. \quad e_{ij} = \text{a}\ (\text{W}\vec{h}_i, \text{W}\vec{h}_j)$$

Where :

a is the weight vector of the attention mechanism

$\vec{h}_i, \vec{h}_j$ are the transformed feature vectors of nodes i and j

W is the weight matrix of the linear transformation

   b. Attention Scores: The softmax function is then used to normalize the attention coefficients in order to produce a probability distribution for node i's neighbors. The model can then focus on the nodes that are the most important for each node.

$$4. \quad \alpha_{ij} = \text{softmax}\ (e_{ij}\ ) = \frac{\exp(e_{ij})}{\sum_{k_i \in N} \exp(e_{ik})}$$

Where:

$\boldsymbol{\alpha_{ij}}$is the attention score for the edge (i,j)

$\boldsymbol{e_{ij}}$ is the attention coefficient between nodes i and j

Attention mechanism is a single layer neural networks, the input for this network are the two transform node features vectors for an edge, and applying the LeakyReLU nonlinearity (with negative input slope α = 0.2). the output indicates the importance between these nodes so the attention coefficients, and applying the LeakyReLU nonlinearity (with negative input slope α = 0.2).

5. $$\alpha_{ij} = \text{softmax}\,(e_{ij}) = \frac{\exp(\text{LeakyReLU}(\vec{a}^{T}[W\vec{h}_i || W\vec{h}_j]))}{\sum_{k_i \in N} \exp(\text{LeakyReLU}(\vec{a}^{T}[W\vec{h}_i || W\vec{h}_k]))}$$

4. *Aggregation and Update Layer:* By aggregating the embeddings of node i's neighbors, weighted by their attention scores, the final embedding vector can be Updated. This is done by using a weighted and nonlinearity sum operation σ.

$$\vec{h'}_i = \sigma \sum_{J \in N_i} (\alpha_{ij}\, W\vec{h}_j)$$

Where :

$\vec{h'}_i$ is the final embedding vector of node i

σ is the activation function

$\alpha_{ij}$is the attention score for the edge (i,j)

|| is the concatenation

W is the weight matrix of the linear transformation

of (potentially, with a different cardinality F'), h'= $\{\vec{h'}_1, \vec{h'}_2, \vec{h'}_3, \ldots, \vec{h'}_N\}$.

5. *Next-item recommendation layer:* The final embedding vector with final scores are used to presents a ranked list of next-item recommendations.

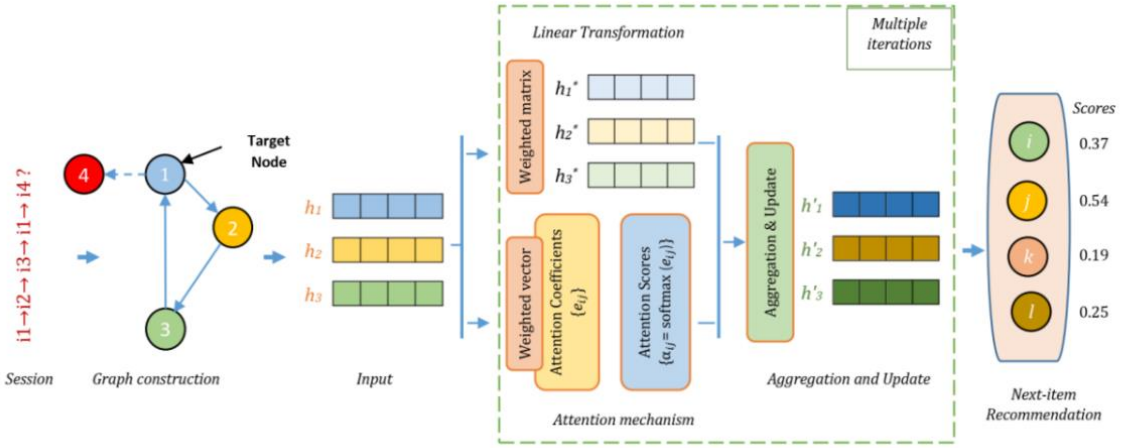


Figure 23: Functioning process of GAT based-model for SBRS

3.4 Material environment

The experiments were conducted on a computer system with the following specifications:

- Processor: Intel Core i5 6th generation
- RAM: 8 GB
- Graphics Card: Integrated graphics
- Storage: Solid State Drive (SSD)

3.5 Development tools and Libraries

The development and implementation of the model were carried out using the following tools and libraries:

1.5.1 Jupyter

Jupyter [1] is an open source web application for creating and sharing interactive notebooks. Released in 2014 and derived from IPython. Jupyter allows combine code, visualizations, and text into one document. It supports multiple programming languages and facilitates reproducible research. Jupyter enables users to iteratively develop code, analyze data, and communicate insights. The ecosystem includes JupyterLab and JupyterHub for advanced functionality and collaboration. The user-friendly interface, language flexibility, and active community make Jupyter a valuable tool in data science and research.

Figure 24: Jupyter logo

3.5.2 Python

Python [2] is a powerful, high-level, interpreted programming language. Designed by Guido van Rossum in 1991, it is known for its simplicity and readability. Python's simple and short syntax places great emphasis on code readability, making it ideal for beginners. It is compatible with various programming paradigms, including procedural, object-oriented, and functional programming. Due to its rich ecosystem of libraries and frameworks, Python is popular in web development, data analysis, machine learning, and automation. Its widespread adoption and continued expansion is supported by extensive community support and documentation. Overall, Python is a powerful and easy-to-use programming language that continues to grow and prosper.

[1] Project Jupyter | Home
[2] Welcome to Python.org

Figure 25: Python programming language logo

3.5.3 PyTorch

PyTorch [3] is an open source machine learning library for Python. It was developed by Facebook's AI research lab and released in 2016. PyTorch provides a dynamic arithmetic framework that enables efficient tensor computation and automatic differentiation. It is widely used to build and train neural networks, especially in deep learning applications. PyTorch's intuitive and flexible design, as well as its extensive collection of pre-built modules and utilities, make it a popular choice among researchers and practitioners. With an active community and constant development, PyTorch remains at the forefront of the deep learning ecosystem, enabling users to accelerate research and build cutting-edge machine learning models.

Figure 26: PyTorch logo

3.5.4 Numpy

NumPy is the fundamental library for numerical computation in Python. It provides support for large multidimensional arrays and matrices, and a collection of mathematical functions for manipulating these arrays. First released in 2006, NumPy has grown to become a cornerstone of scientific computing and data analysis. It provides efficient data structures, transfer functions, and vectorized operations to improve the performance of numerical calculations. Due to its rich functionality and integration with other scientific libraries, NumPy is widely used in fields such as data analysis, machine learning, and scientific research, enabling users to process and manipulate numerical data efficiently.

[3] PyTorch

Figure 27: Numpy logo

3.5.5 Scikit-learn

Scikit-learn is a powerful machine learning library for Python. It provides a comprehensive set of tools for various machine learning tasks, including classification, regression, clustering, and dimensionality reduction. First released in 2007, Scikit-learn has grown to become the go-to resource for machine learning practitioners and researchers. It provides a user-friendly interface, extensive documentation, and a wide range of algorithms and techniques. Scikit-learn integrates seamlessly with other scientific libraries in the Python ecosystem, making it a versatile choice for data analysis and model development. With its powerful features and community support, scikit-learn enables users to efficiently explore, model and solve complex machine learning problems.

Figure 28: Scikit-learn logo

There are countless additional libraries and frameworks available in Python for a variety of uses. Here are a few famous examples:

- Torch_geometric: A PyTorch library for geometric deep learning. It provides modules for constructing graph data structures (for example, Data), graph operations (for example, to undirected), and neural network layers for graph data (for example GATConv).
- Pandas (imported as pd): A data manipulation and analysis library.
- Unbalanced-learn (imblearn): A library for dealing with imbalanced datasets that includes oversampling (e.g., RandomOverSampler) and undersampling (e.g. RandomUnderSampler) approaches.

3.6 Experiments and implementations

3.6.1 Evaluation metrics

The performances of the proposed model and compared methods are evaluated by

- Accuracy : The percentage of cases that were successfully predicted out of all instances is used to calculate accuracy. The number of accurate forecasts divided by the total number of predictions is used to compute it. Accuracy, however, might not be the best statistic for datasets with imbalances.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

- Precision is the percentage of accurate positive predictions compared to all positive forecasts. It measures the proportion of true positives to the total of true positives and false positives and focuses on the accuracy of positive forecasts.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

- Recall (Sensitivity or True Positive Rate): Recall counts the number of actual positive cases out of all the true positive forecasts. It is measured as the ratio of true positives to the total of true positives and false negatives and focuses on the model's capacity to detect positive cases.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

- F1-Score: The harmonic mean of recall and precision is the F1 score. When you wish to take both the precision and recall values into account, it offers a balanced measurement between the two. It is determined by dividing the sum of precision and recall by two times the product of precision and recall.

$$\text{F1 Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

Where TP represents the number of correct recommendations, TN represents the number of correctly identified non-relevant items, and FP indicates the number of incorrect recommendations. FN indicates the number of desired items that are not included in the recommendation list.

3.6.2 Datasets

The measurements made for datasets are from two different domains: e-commerce and movies

- **E-Commerce Datasets:** the following e-commerce dataset is used

1. RetailRocket [4]: A publicly accessible e-commerce dataset called Retail Rocket covers user interactions with online retail platforms. It is frequently employed in the study and assessment of recommendation systems. The collection is made up of anonymous user behaviors including add-to-cart, buy, and view actions. Each event has a timestamp, user session ID, and object ID attached to it.
2. Yoochoose [5] : The YooChoose dataset is another publicly available e-commerce dataset commonly used for recommendation system research and evaluation. It contains user click data from an online retail platform. The dataset captures user interactions such as clicks, views, and purchases, along with the associated session information.
   The YooChoose dataset is particularly suitable for session-based recommendation tasks, where the goal is to recommend items based on the user's current session or recent browsing history. It provides a valuable resource for studying user behavior, session dynamics, and developing personalized recommendation algorithms.

- **Media Datasets:**

1. MovieLens100K[6]
   The MovieLens 100K dataset is a widely used benchmark dataset in the field of recommender systems. It contains movie ratings provided by users of the MovieLens website. The dataset

[4] Available at: Retailrocket recommender system dataset | Kaggle
[5] Available at: yoochoose | Kaggle
[6] Available at: MovieLens 100K Dataset | Kaggle

is commonly used for tasks such as collaborative filtering, where the goal is to predict user ratings for movies based on historical ratings and user preferences.

The statistics for the datasets are given in Table :

	RetailRocket	Yoochoose	MovieLens
Rows	35309	74708	100226
Sessions	3636	14514	22202
Items	15144	9255	9723

Table 4: Characteristics of the datasets

3.6.3 Loss functions

While using GAT to address session-based or more general sequential prediction problems is a natural choice, the specific network architecture, choice of attention mechanism, and use of session-parallel mini-batches to speed up the training phase are key innovative elements of the technique. method. The GAT architecture is able to capture complex relationships and dependencies between elements in a session using an attention mechanism, which proves to be well suited for modeling sequential data. By assigning different importance weights to neighboring nodes, GAT selectively focuses on related items, allowing efficient learning and representation of the importance of different items in a single session. Furthermore, the choice of attention mechanism in GAT demonstrates its ability to handle complex dependencies within session data, making it an effective solution for session-based prediction tasks.

Typically, there are several hyper-parameters that can be adjusted, such as the learning rate, in_features, out_features, num_heads, and a dropout factor that helps maintain network stability. Another essential aspect determining the effectiveness of SBRS with GAT is the selection of the loss function.

- The method of ***Cross-Entropy Loss*** is frequently employed in architectural work to evaluate the difference between the probabilities of the predicted class and the target class. In this case, the predicted class probabilities are derived from the GAT model's output, whereas the target class is the subsequent item in each session. The loss function works by computing the negative logarithmic-likelihood of the anticipated class probabilities concerning the target class. And it is defined with the following equation:

$$L_{CE} = -\sum_{i=1}^{n} t_i log_{(p_i)}$$

For n classes, where t_i is the truth label and p_i is the Softmax probability for the i^{th}class.

And the hyperparameters include:

1. In_features: The input feature dimension of each node in the graph. In this case, it is set to 1 because node characteristics are represented by the position of each element in the session.

2. Out_features: The output feature dimension of each node after applying GAT convolution. In this code, it is set to 100.

3. Num_heads: The number of attention heads used in GAT convolution. It controls how often the attention mechanism is applied. The code uses 8 attention heads.
4. Dropout: The dropout rate applied after GAT convolution to prevent overfitting. Setting it to 0.1 means that 10% of node features are randomly set to zero during training.

3.6.4 Implementation

The code performs a session-based recommendation using a Graph Attention Network (GAT). The implementation can be summarized as follows:

1) The necessary libraries are imported, including pandas, numpy, scikit-learn, torch, and imbalanced-learn.

```
#import libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import torch
from torch_geometric.data import Data
from torch_geometric.utils import to_undirected
import torch.nn as nn
import torch.nn.functional as F
from torch_geometric.nn import GATConv
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from torch.utils.data import DataLoader
```

Figure 29: Import necessary libraries

2) The data is read from a CSV file and preprocessed. Irrelevant columns are dropped, and the timestamp is converted to datetime format. The data is sorted based on visitor ID and timestamp. The previous item ID for each visitor is added as a new column. Any rows with missing values are dropped.

```
data = pd.read_csv('/content/drive/MyDrive/retailrocket/events.csv' , nrows = 80000)
# Drop the 'rating' column
data = data.drop('event', axis=1)
data = data.drop('transactionid', axis=1)
# The pd.to_datetime() function is used to convert the 'timestamp' column of the dataset
# to a datetime format. The unit='s' argument specifies that the timestamp values are in seconds.
data['timestamp'] = pd.to_datetime(data['timestamp'] / 1000, unit='s')
data = data.sort_values(['visitorid', 'timestamp'])
data['prev_item_id'] = data.groupby('visitorid')['itemid'].shift(1)
data = data.dropna()
```

Figure 30: Preprocessing the data

3) The data is split into training and testing sets using a train-test split with a 80:20 ratio.

```
# Split the train data into training and testing sets
train_data, test_data = train_test_split(data, test_size=0.2)
```

Figure 31: Split the dataset

4) Item IDs are mapped to indices. Unique item IDs from both the training and testing sets are combined, and each item is assigned a unique index using a dictionary.

```
# Map item IDs to indices
train_items = train_data['itemid'].unique()
test_items = test_data['itemid'].unique()
all_items = np.union1d(train_items, test_items)
num_items = len(all_items)
item_to_idx = {item: idx for idx, item in enumerate(all_items)}
train_data['item_idx'] = train_data['itemid'].map(item_to_idx)
test_data['item_idx'] = test_data['itemid'].map(item_to_idx)
```

Figure 32: Mapping the Item IDs to indices

5) Out-of-range indices are filtered out, and sessions are constructed for the training and testing data. Sessions are grouped by visitor ID, and each session contains a sequence of item indices.

```
train_data = train_data[train_data['item_idx'] < num_items]
test_data = test_data[test_data['item_idx'] < num_items]
```

Figure 33: Filtering any out of range indices

```
# Construct sessions for training data
train_sessions = []
for user_id, group in train_data.groupby('visitorid'):
    session = []
    prev_timestamp = None

    for _, row in group.iterrows():
        if prev_timestamp is None or row['timestamp'] - prev_timestamp <= time_threshold:
            session.append(row['item_idx'])  # Append 'item_idx' directly
        else:
            if len(session) >= min_session_length and len(session) <= max_session_length:
                train_sessions.append(session)
            session = [row['item_idx']]  # Start a new session
        prev_timestamp = row['timestamp']

    if len(session) >= min_session_length and len(session) <= max_session_length:
        train_sessions.append(session)

# Construct sessions for testing data
```

Figure 34: Construct sessions for training and testing data

6) The lengths of the sessions in the training and testing sets are calculated.
7) Dynamic graphs are created for the training and testing data. The graphs consist of node features, edge indices, edge attributes, and edge timestamps. The number of nodes is equal to the total number of unique items.

```
# Create the train graph as a dynamic graph
train_graph = Data(
    x=train_node_features,
    edge_index=train_edge_index,
    edge_attr=train_edge_attr,
    edge_time=train_edge_time,
    num_nodes=num_items
)
```

Figure 35: Create the dynamic train graph

8) A Graph Attention Network (GAT) model is defined using the torch_geometric library. The GAT model is implemented as a subclass of the nn.Module class and includes the necessary layers and functions.

```
import torch.nn.functional as F

class GAT(nn.Module):
    def __init__(self, in_features, out_features, num_heads, dropout=0.1):
        super(GAT, self).__init__()
        self.conv = GATConv(
            in_channels=in_features,
            out_channels=out_features,
            heads=num_heads,
            dropout=dropout
        )
        self.fc = nn.Linear(out_features * num_heads, num_items)
        self.leaky_relu = nn.LeakyReLU(0.2)  # Initialize LeakyReLU activation function with negative slope 0.2

    def forward(self, data):
        x, edge_index, edge_attr = data.x, data.edge_index, data.edge_attr
        num_nodes = x.size(0)
        # Filter out any indices that are out of bounds
        mask = (edge_index < num_nodes).all(dim=0)
        edge_index = edge_index[:, mask]
        edge_attr = edge_attr[mask]
        x = self.conv(x, edge_index, edge_attr)
        x = x.view(-1, self.conv.out_channels * self.conv.heads)
        x = self.fc(x)
        return F.log_softmax(x, dim=-1)  # Use log_softmax for classification
```

9) The GAT model is trained and evaluated. An Adam optimizer is used with a learning rate of 0.01, and the cross-entropy loss is used as the loss function. The training loop iterates over a specified number of epochs. In each epoch, the model is trained using the training graph, and then evaluated using the testing graph. Training and testing loss, as well as accuracy, are computed and can be tracked.

This implementation utilizes session data, constructs dynamic graphs, and applies a GAT model for session-based recommendation. It provides a framework for training and evaluating the GAT model on the given data.

Note: In the implementation, the Graph Attention Network (GAT) model utilizes a dynamic graph representation. A dynamic graph consists of time-varying relationships between nodes, where the edges in the graph are associated with timestamps and attributes. This allows the model to capture temporal dependencies and consider the order of interactions between nodes.

However, it is important to note that handling dynamic graphs can consume a significant amount of memory, particularly when dealing with large datasets. Due to memory constraints, in this

implementation, a smaller dataset is used to avoid potential session crashes or out-of-memory errors, particularly when running the code in an environment like Google Colab. By using a smaller dataset, the memory usage is reduced, enabling the successful execution of the model without encountering memory-related issues.

3.7 Results

Table 5 shows the results of the proposed GAT-based SBRS model on the aforementioned evaluation datasets using the accuracy metric. Several architectures were examined and a single layer of GAT was found to be the best performer.

Epochs	RETAILROCKET	YOOCHOOSE	MOVIELENS
100	0.7923	0.6158	0.5751
200	0.7933	0.6165	0.5743

Table 5: The results of the proposed GAT-based SBRS model

3.7.1 Comparison with Baselines

POP Baseline: also known as the popularity baseline, is a simple recommendation approach that suggests items to users based on their overall popularity or frequency of occurrence in a dataset.

S-POP Baseline: In session-based recommendation systems, where user interactions are captured in sequential sessions or sessions of user activities, S-Pop refers to a popularity measure that takes into account the popularity of items within a specific session. It focuses on recommending items that are popular within the current session.

Random Recommendations : Random recommendation baseline is a simple and straightforward approach in recommendation systems where items are selected randomly and recommended to users. It serves as a basic benchmark for evaluating the performance of more sophisticated recommendation algorithms.

The table allows comparisons between the three baselines with the outcomes of the GAT-based SBRS model with metrics (precision, recall, and F1-score), it clearly shows that our model dominates the other methods.

Baselines	RETAILROCKET			YOOCHOOSE		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
POP	0.4546	0.6462	0.5333	0.5342	**0.9241**	0.6771
S-POP	0.0792	0.2229	0.1169	0.1106	0.1944	0.1410
Random recommendations	0.0721	0.2920	0.1156	0.1014	0.1842	0.1308
GAT-based SBRs model	**0.7084**	**0.7807**	**0.7428**	**0.9505**	**0.6036**	**0.7384**

Table 6: Comparison of GAT-based SBRS model with the baselines

Based on the previous results, we can draw a comparison between the proposed model and the baselines using these graphs, which vividly illustrate the performance differences.

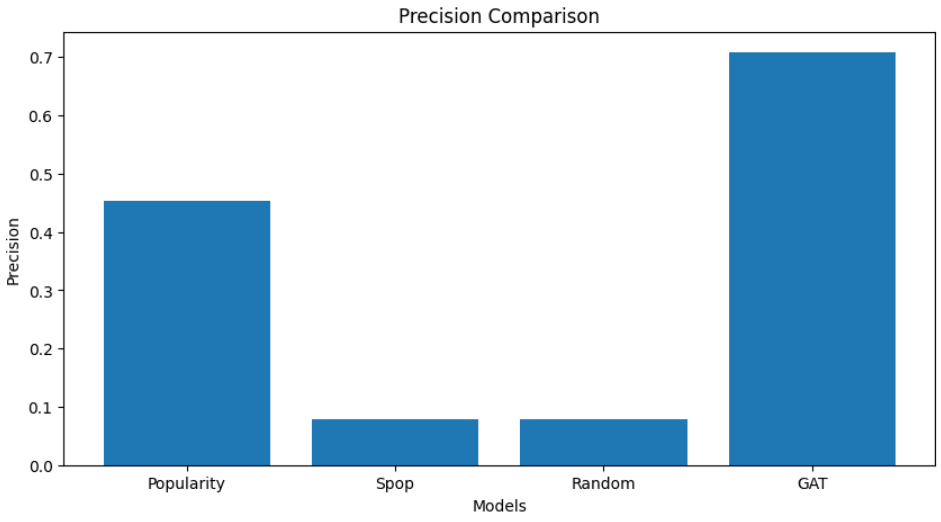


Figure 36: Model performance comparison with RETAILROCKET dataset using precision metric

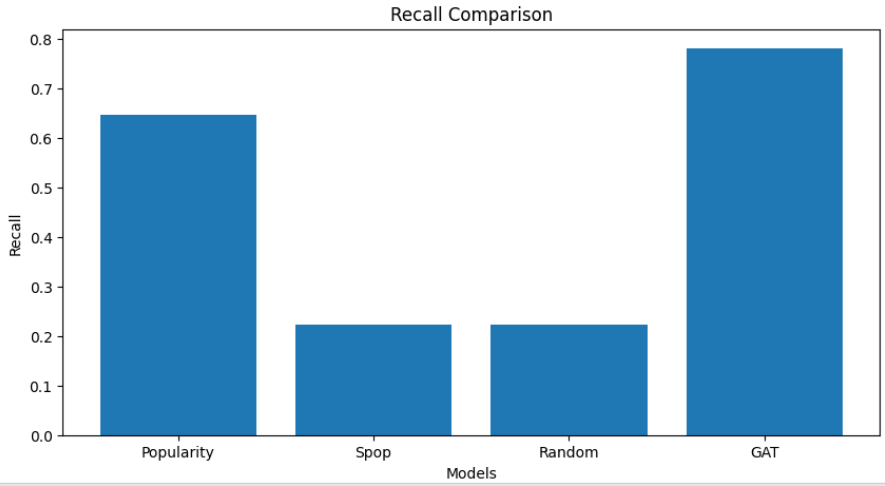


Figure 37: Model performance comparison RETAILROCKET dataset using recall metric

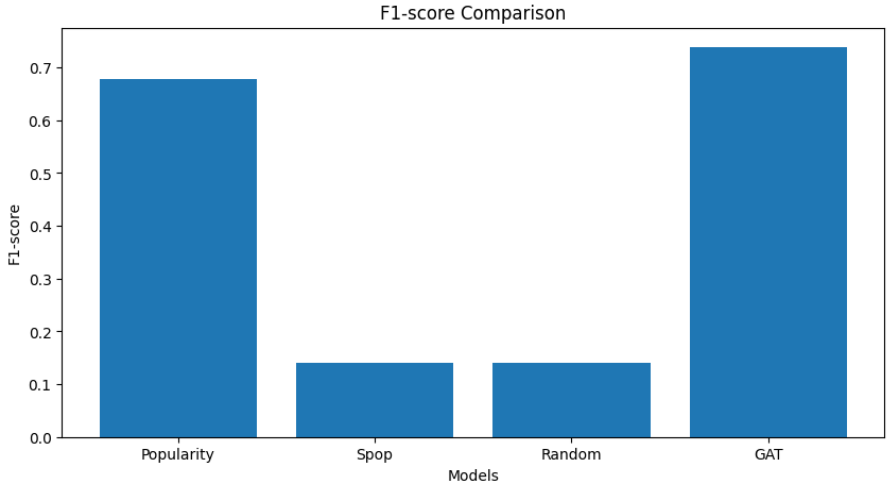


Figure 38: Model performance comparison YOOCHOOSE dataset using F1-score metric

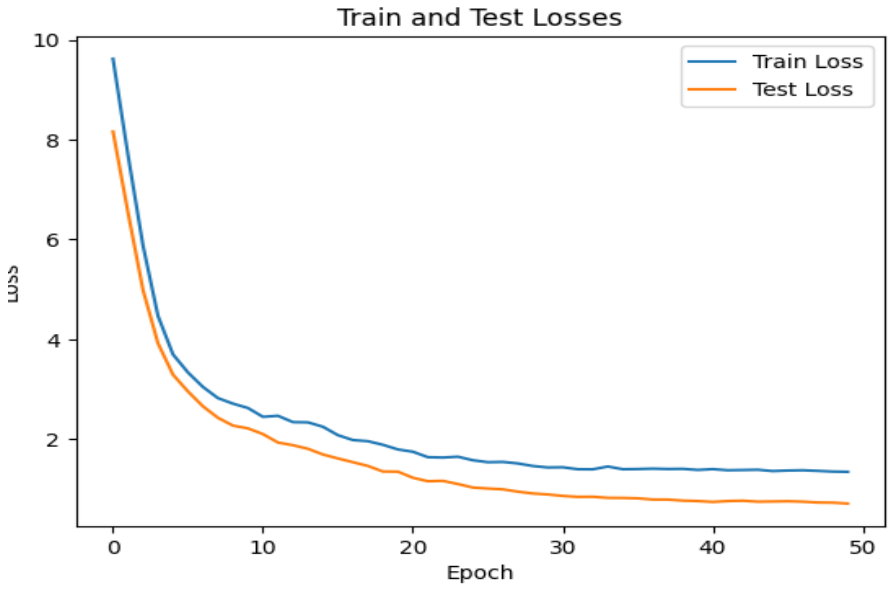


Figure 39: Train and test losses for RETAIL-ROCKET dataset

3.7.3 Discussion

In this study, we compared the performance of our GAT-based SBRS (Session-Based Recommendation System) model with three baselines (POP, S-POP and Random recommendations) on the RETAILROCKET and YOOCHOOSE datasets. The evaluation was based on precision, recall, and F1-score metrics.

The results clearly demonstrate the superiority of our GAT-based SBRS model over the baselines. On the RETAILROCKET dataset, our model achieved a precision of 0.7084, a recall of 0.7807, and F1-score of 0.7428. Similarly, on the YOOCHOOSE dataset, our model achieved a precision of 0.9505, recall of 0.6036, and F1-score of 0.7384. In comparison, the baselines exhibited lower performance across all metrics.

The performance improvements of our GAT-based SBRS model can be attributed to several factors. Firstly, the GAT architecture employed in our model has the ability to capture complex relationships and dependencies among items in sessions. By leveraging attention mechanisms, the model assigns higher weights to important items in the session, thereby enabling more accurate recommendations. This is particularly beneficial in session-based recommendation scenarios where user preferences can change dynamically.

Furthermore, the inclusion of temporal edge indices, edge attributes, and edge times in our model enhances the understanding of session dynamics. By considering the order of items, their attributes, and the timing of interactions, our model gains a deeper insight into the evolving preferences and interests of users. This additional contextual information aids in making more personalized and relevant recommendations.

The practical implications of our GAT-based SBRS model are significant. By providing accurate session-based recommendations, e-commerce platforms can enhance the user experience and increase user engagement. Personalized and relevant recommendations can lead to improved customer satisfaction, potentially resulting in higher conversion rates and revenue for retail companies. Our model's performance improvements offer a promising avenue for businesses to deliver more effective recommendations and drive user engagement.

Despite the promising results, it is important to acknowledge some limitations of our study. Firstly, the evaluation was performed on two specific datasets, RETAILROCKET and YOOCHOOSE. While these datasets are widely used in the field, they may not fully represent the diversity of real-world scenarios. Future research should consider evaluating the model on additional datasets to validate its performance across different contexts.

In terms of future directions, there are several avenues to explore. One potential area for improvement is the incorporation of additional contextual information, such as user demographics, item attributes, or session context. By leveraging a wider range of information, the model can gain a deeper understanding of user preferences and provide even more accurate recommendations. Additionally, exploring alternative graph-based architectures or integrating other deep learning techniques could further enhance the performance of session-based recommendation systems.

In conclusion, our study demonstrates the effectiveness of the GAT-based SBRS model in session-based recommendation scenarios. The model's superior performance compared to the baselines and

its competitive standing in relation to existing approaches validate its potential contributions to the field. The practical implications of accurate session-based recommendations emphasize the value of our model in improving user experience and driving business outcomes. Further research and advancements in this area will continue to refine and extend the capabilities of session-based recommendation systems.

3.8 Conclusion

At the beginning of this chapter, the Deep learning-based recommendations were discussed. The focus then shifted towards the use of one of the deep learning models: GATs on session-based recommendations, which are becoming one of the most important recommendation approaches in practice for many domains, including movies, and general e-commerce. A GAT Based-Model was proposed for SBRS and proved to perform better than the three baselines used (pop, s-pop, random recommendations) in terms of the three-evaluation metrics introduced: precision, recall and F1.

General Conclusion

4. General conclusion

A. Summary

First, a session-based recommender System was reported in *Chapter 1* by a survey that was conducted to determine the recommendation system, the problem of a session-based recommender, and the position of session-based in the family of sequence-aware recommendations by a comparison of SBRS and other RSs. Also, categorization, approaches, challenges, and limitations concluded it.
Then, *in Chapter 2*, deep learning is studied in detail from the basics, namely H. Neural Networks, Backpropagation, Gradient Descent, Loss, and Activation Functions to Address the Challenge of Employing Deep Learning Approaches Representing Graph Attention Networks in Session-Based Recommendation.
In *Chapter 3*, in order to answer the research question, a recommender system is created, refined, and evaluated based on the session-based model. The end result of this work is a consideration of research findings and their implications.
The research questions discussed are:

“"How can a Graph Attention Network (GAT) model be effectively utilized in session-based recommendation systems to capture the intricate and dynamic patterns of user behavior within individual sessions, leading to improved and personalized recommendations?"

Session-based recommendation using GAT is now widely recognized as one of the primary means of recommendation in many areas, To enhance the performance of session-based recommendation systems (SBRS), With the proposed GAT-Based-Model approach, it was demonstrated how Graph ATtention Networks can incorporate sessions in recommender systems, which provides superior results on the three primary evaluation metrics (precision, recall, and F1) compared to the four baselines employed (pop, spop, Random, and popularity). on three datasets, each from a different domain (e-commerce and music).
It is also relevant to note that a part of this research has been published in the RIA’2023 Conference (for more information, refer to the appendix at the end of this document).

B. Directions for future research

From this survey, and based on the results of GAT shows in the field of Session-based recommender systems, we plan to perform the following research:

1. Enhancing Graph Attention Mechanisms:
    - Investigate innovative ways to increase the effectiveness of graph attention mechanisms in session-based recommendation systems.
    - Explore other attention mechanisms, such as self-attention or multi-head attention, to capture more complicated interactions within the graph.
2. Incorporating Temporal Dynamics:
    - Study techniques to include temporal dynamics into session-based recommendation systems using graph attention networks.

- Develop methods for modeling the evolution of user preferences over time and modify the recommendations accordingly.

3. Evaluation Metrics and Benchmarks:
    - Develop standardized evaluation standards and benchmark datasets particularly tailored for session-based recommendation systems leveraging graph attention networks.
    - Enable fair comparisons between different techniques and encourage improvements in the field.
4. Real-world Application: Job Recommendation Website under the name of *"KHADEMNI.DZ".*

Prototype

5. Prototype

In the future, we plan to launch a job recommendation website called "KHADAMNI.DZ."

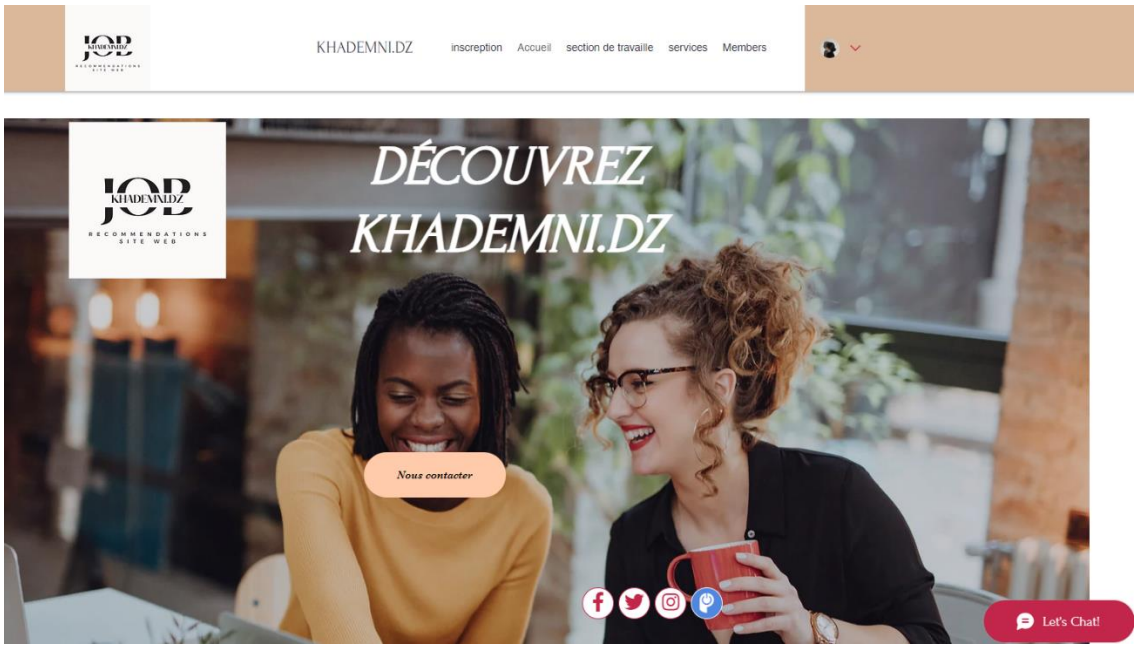


Figure 40: Home page of the job recommendation website

- The job recommendation website "KHADEMNI.DZ" uses graph attention network architecture to provide personalized job recommendations based on user session data. Even users who are new to the platform or have limited browsing history can benefit from tailored recommendations ,This website offers two modes of interaction: one through registration or inscription, and the other in an anonymous manner. The site's recommendation system analyzes user sessions for preferences and interests to ensure accurate and relevant job recommendations.

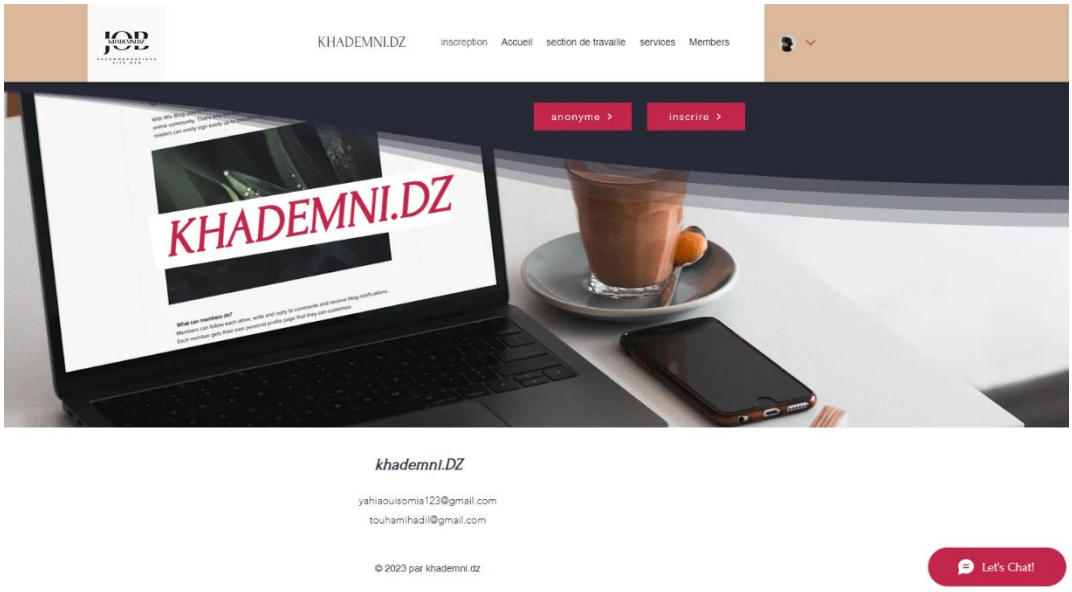


Figure 41 :Exploring the Dual Pathways Registration and Anonymity in Website Interaction

- The site provides job postings from various industries, ensuring users have access to a wide range of employment opportunities. Covering industries such as technology, medical care, finance, and education, users can explore employment opportunities in their favorite industries.

Figure 42: Job postings

- The job recommendation website "KHADEMNI.DZ" employs a graph attention network architecture to deliver job recommendations primarily based on user session interactions within the platform. By examining user behavior during each session, such as job advertisements viewed and jobs applied for, the site generates personalized suggestions that cater to the user's immediate interests and activities. This approach ensures that the job recommendations provided align closely with the user's recent session interactions, maximizing relevance and enhancing the overall user experience.

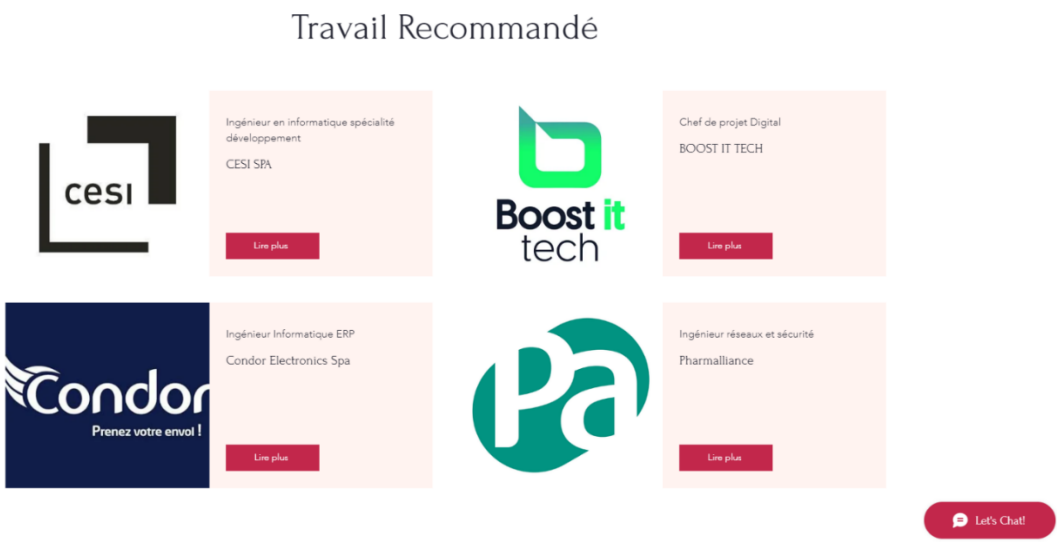


Figure 43: Job recommendation

The development process involved designing and building a web platform integrating the graph attention network approach. The method leverages the power of graph neural networks to analyze and understand relationships among various job attributes such as job title, skills, and industry. By taking these relationships into account, the system can generate more accurate and relevant recommendations.

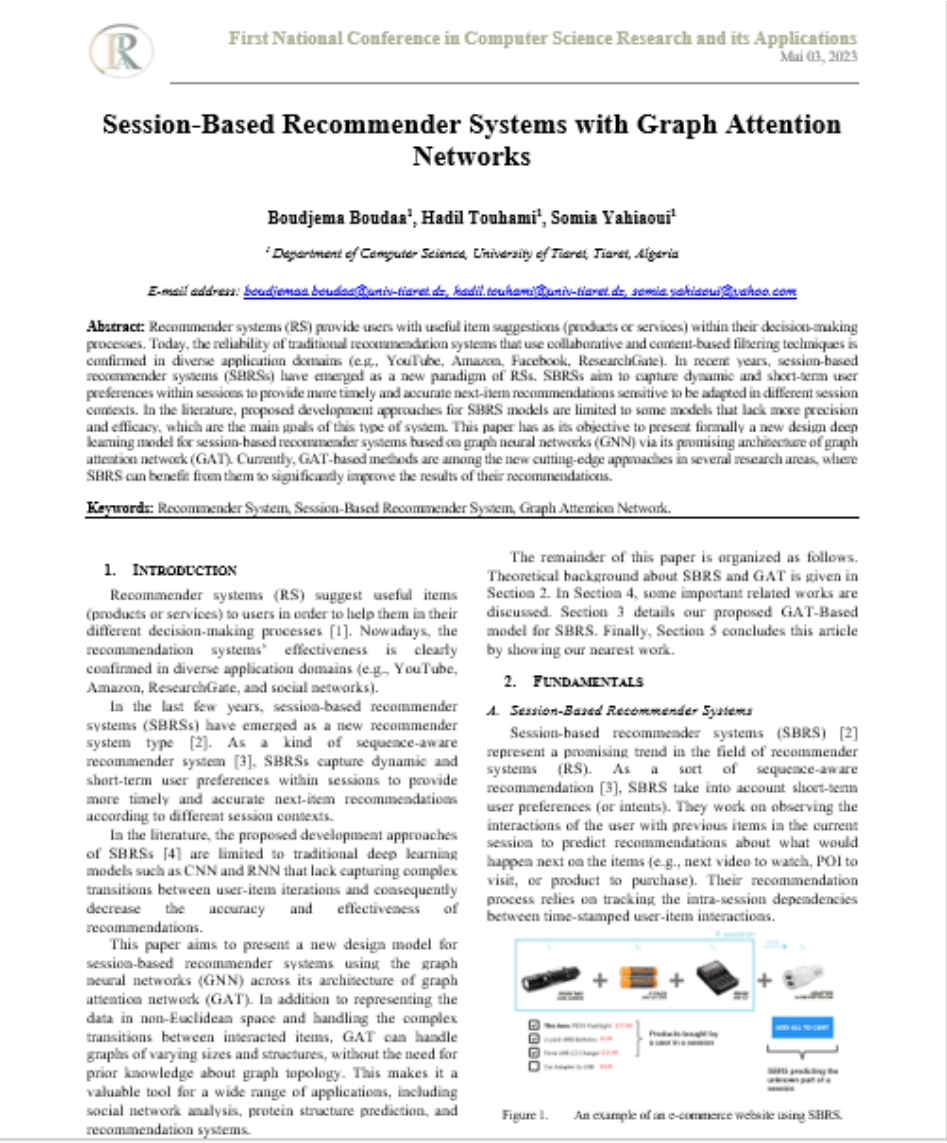
Appendix

6. Appendix

The following appendix presents an article on a novel session-based recommendation system utilizing Graph Attention Networks (GAT) named "*Session-Based Recommender Systems with Graph Attention Networks*" . This article, published in RIA23 the *First National Conference in Computer Science Research and its Applications* , focuses on the design and architecture of the proposed model, aiming to address the challenges associated with session-based recommendations.

The article provides an in-depth description of the GAT-based architecture, outlining the key components and their functionalities. We delve into the attention mechanism employed by GAT and explain how it enables the model to dynamically weigh the importance of individual items based on their relevance to the current session context.

Note: that the following content represents only the first page of the article. This limitation is imposed to respect copyright laws and restrictions.



First National Conference in Computer Science Research and its Applications
Mai 03, 2023

Session-Based Recommender Systems with Graph Attention Networks

Boudjema Boudaa[1], Hadil Touhami[1], Somia Yahiaoui[1]

[1] *Department of Computer Science, University of Tiaret, Tiaret, Algeria*

E-mail address: boudjemaa.boudaa@univ-tiaret.dz, hadil.touhami@univ-tiaret.dz, somia.yahiaoui@yahoo.com

Abstract: Recommender systems (RS) provide users with useful item suggestions (products or services) within their decision-making processes. Today, the reliability of traditional recommendation systems that use collaborative and content-based filtering techniques is confirmed in diverse application domains (e.g., YouTube, Amazon, Facebook, ResearchGate). In recent years, session-based recommender systems (SBRSs) have emerged as a new paradigm of RSs. SBRSs aim to capture dynamic and short-term user preferences within sessions to provide more timely and accurate next-item recommendations sensitive to be adapted in different session contexts. In the literature, proposed development approaches for SBRS models are limited to some models that lack more precision and efficacy, which are the main goals of this type of system. This paper has as its objective to present formally a new design deep learning model for session-based recommender systems based on graph neural networks (GNN) via its promising architecture of graph attention network (GAT). Currently, GAT-based methods are among the new cutting-edge approaches in several research areas, where SBRS can benefit from them to significantly improve the results of their recommendations.

Keywords: Recommender System, Session-Based Recommender System, Graph Attention Network.

1. Introduction

Recommender systems (RS) suggest useful items (products or services) to users in order to help them in their different decision-making processes [1]. Nowadays, the recommendation systems' effectiveness is clearly confirmed in diverse application domains (e.g., YouTube, Amazon, ResearchGate, and social networks).

In the last few years, session-based recommender systems (SBRSs) have emerged as a new recommender system type [2]. As a kind of sequence-aware recommender system [3], SBRSs capture dynamic and short-term user preferences within sessions to provide more timely and accurate next-item recommendations according to different session contexts.

In the literature, the proposed development approaches of SBRSs [4] are limited to traditional deep learning models such as CNN and RNN that lack capturing complex transitions between user-item iterations and consequently decrease the accuracy and effectiveness of recommendations.

This paper aims to present a new design model for session-based recommender systems using the graph neural networks (GNN) across its architecture of graph attention network (GAT). In addition to representing the data in non-Euclidean space and handling the complex transitions between interacted items, GAT can handle graphs of varying sizes and structures, without the need for prior knowledge about graph topology. This makes it a valuable tool for a wide range of applications, including social network analysis, protein structure prediction, and recommendation systems.

The remainder of this paper is organized as follows. Theoretical background about SBRS and GAT is given in Section 2. In Section 4, some important related works are discussed. Section 3 details our proposed GAT-Based model for SBRS. Finally, Section 5 concludes this article by showing our nearest work.

2. Fundamentals

A. Session-Based Recommender Systems

Session-based recommender systems (SBRS) [2] represent a promising trend in the field of recommender systems (RS). As a sort of sequence-aware recommendation [3], SBRS take into account short-term user preferences (or intents). They work on observing the interactions of the user with previous items in the current session to predict recommendations about what would happen next on the items (e.g., next video to watch, POI to visit, or product to purchase). Their recommendation process relies on tracking the intra-session dependencies between time-stamped user-item interactions.

Figure 1. An example of an e-commerce website using SBRS.

This appendix includes an attestation that was granted to us subsequent to the successful online presentation of our article

جامعة غليزان
RELIZANE UNIVERSITY

CERTIFICATE

OF PARTICIPATION

Presented to

Boudjemaa BOUDAA

For participation in online conference

First National Conference in Computer Science Research and its Applications RIA'23

to be held on May 10, 2023

Session-Based Recommender Systems with Graph Attention Networks

Co-authors: Hadil TOUHAMI, Somia YAHIAOUI

د . جيلالي بكداد
عميد كلية العلوم والتكنولوجيا

Dr. Abdelkrim BOUADJEMI
Chairman of the Conference

Bibliography

Bibliography

[1] G. A. a. A. Tuzhilin, " A survey of the state-of-the-art and possible extensions," *Toward the next generation of recommender systems,* vol. Volume 17(NO. 6), p. Pages 734 _ 749, 2005.

[2] L. C. a. Y. W. oujin Wang, "A survey on session-based recommender systems," 2019.

[3] L. Y. A. S. a. Y. T. Shuai Zhang, "A survey and new perspectives," *Deep learning based recommender system,* vol. Volume 52(NO. 1), 2019.

[4] A. I. C. D. L. S. M. S. K. J. e. a. Rashid AM, "Getting to know you: learning new user preferences in recommender systems.," p. 127–34., 2002.

[5] K. J. R. J. Schafer JB, "Recommender system in e-commerce," in *Proceedings of the 1st ACM conference on electronic commerce*, 1999.

[6] H. V. P. Resnick, "Recommender system's," 1997.

[7] A. A. A.M. Acilar, "A collaborative filtering method based on Artificial Immune Network," pp. 8324-8332, 2009.

[8] "Nvidia," [Online]. Available: https://www.nvidia.com/en-us/glossary/data-science/recommendation-system/.

[9] C. S. J. Buder, "Learning with personalized recommender systems: a psychological view," pp. 207-216, 2012.

[10] L. N. Gadanho SC, "Addressing uncertainty in implicit preferences.," New York, NY, USA, 2007.

[11] "Hybrid Recommendation System – A Beginner's Guide," [Online]. Available: https://www.muvi.com/resources/ebooks/hybrid-recommendation-system.

[12] O. O. Poddubnyy, "GRAPH NEURAL NETWORKS FOR RECOMMENDER SYSTEMS," 2021.

[13] R. B.-Y. a. B. Ribeiro-Neto., "Modem Information Retrieval," p. 271–350, 1999.

[14] F. C. Z. L. Hui Li, "Content-Based Filtering Recommendation Algorithm Using HMM," in *2012 Fourth International Conference on Computational and Information Sciences*.

[15] B. S. a. J. Y. Greg Linden, "recommendations: Item-to-item collaborative filtering.," 2003.

[16] Burke, "R. Hybrid Recommender Systems," *User Model User-Adap Inter 12,* p. 331–370, 2002.

[17] "analyticsindiamag," 2021. [Online]. Available: https://analyticsindiamag.com/cold-start-problem-in-recommender-systems-and-its-mitigation-techniques/.

[18] L. S. a. A. Gera., "A Survey of Recommendation System:Research Challenges," *International Journal of Engineering Trends andTechnology (IJETT),* pp. 1989-1992, 2013.

[19] Z. SOUHILA, "Session-based recommender systems using recurrent neural network," 2020.

[20] Z. A. a. I. U. Shah Khusro, "Recommender Systems: Issues, Challenges,and Research Opportunities," February 2016.

[21] F. O. A. H. a. A. G. J. Bobadilla, "Recommender systems survey. Knowledge-Based Systems," vol. 46, p. 109_132, 2013.

[22] D. J. ,. C. Massimo Quadrana, "Tutorial: Sequence-Aware Recommender Systems," in *WWW '19: Companion Proceedings of The 2019 World Wide Web Conference*, May 2019.

[23] P. C. J. Massimo Quadrana, "Sequence-Aware Recommender Systems," February 2018.

[24] L. C. a. Y. W. Shoujin Wang, " A survey on session-based recommender system," 2019.

[25] H. D. L. N. Bamshad Mobasher, "Effective personalization based on association rule discovery from web usage data," in *Proceedings of the 3rd international workshop on Web information and data management*, 2001.

[26] I. I. B. N. H. D. K. &. J. V. Fabian Abel, "A Rule-Based Recommender System for Online Discussion Forums," 2008.

[27] R. B. V. S. &. V. K. Utpala Niranjan, "Developing a Web Recommendation System Based on Closed Sequential Patterns," 2010.

[28] M. V. D. K. Magdalini Eirinaki, "Web Path Recommendations based on Page Ranking and Markov models," in *Proceedings of the 7th annual ACM international workshop on Web information and data management. ACM*, 2005.

[29] C. F. a. L. S.-T. Steffen Rendle, "Factorizing personalized markov chains for next-basket recommendation.," in *In Proceedings of the 19th international conference on World wide web. ACM–*, 2010.

[30] L. R. F. Asnat Greenstein-Messica, "Session-Based Recommendations Using Item Embedding," in *IUI '17: Proceedings of the 22nd International Conference on Intelligent User Interfaces*, March 2017.

[31] Q. Z. H. ,. Z. ,. W. A. Shoujin Wang, "Sequential/Session-based Recommendations: Challenges,Approaches, Applications and Opportunities," ACM, New York, NY, July 11–15, 2022.

[32] Y. B. a. A. C. I. Goodfellow, "Deep learning," *MIT Press,* 2016.

[33] I. Limited, "Artificial Intelligence and Machine Learning In Mobile Apps," [Online]. Available: https://blogs.infosys.com/digital-experience/mobility/artificial-intelligence-and-machine-learning-in-mobile-apps.html. [Accessed 20 May 2023].

[34] V. Lendave, "A Comprehensive Guide to Representation Learning for Beginners," November 4, 2021. [Online]. Available: https://analyticsindiamag.com/a-comprehensive-guide-to-representation-learning-for-beginners/.

[35] S. Ozechi, "Feature Engineering Techniques," sep 10, 2010. [Online]. Available: https://towardsdatascience.com/feature-engineering-techniques-bab6cb39ed7e.

[36] B. S. a. S. Swapna, "“A Comprehensive Overview on Types of Machine Learning," vol. 04, 2015.

[37] S. P. F. C. G. L. C. Z. a. P. S. Y. Z. Wu, "A Comprehensive Survey on Graph Neural Networks," in *IEEE Trans. Neural Netw. Learning Syst*, Jan. 2021.

[38] E. G. PhD, "Deep Learning: Unleashing the Power of Artificial Intelligence," May 18, 2023. [Online]. Available: https://medium.com/@evertongomede/deep-learning-unleashing-the-power-of-artificial-intelligence-603fd3dc8cdc . [Accessed May 20, 2023].

[39] J. Fumo, "A Gentle Introduction To Neural Networks Series — Part 1," Oct. 22, 2017.

[40] G. Rajgopal, "Deep Learning," Sep. 20, 2019.

[41] "How the Backpropagation Algorithm is Used to Train Neural Networks - AITechTrend," Apr. 18, 2023..

[42] "Neural Network Activation Function," 2023. [Online]. Available: https://neuralnetwork101.com/activation-function.

[43] "Activation Functions in Neural Networks [12 Types & Use Cases]," May 31, 2023.

[44] B. Krishnamurthy, "An Introduction to the ReLU Activation Function," Oct. 28, 2022. [Online]. Available: https://builtin.com/machine-learning/relu-activation-function.

[45] "Relu1 And Relu6: Activation Functions With Benefits And Drawbacks," November 19, 2022. [Online]. Available: https://www.surfactants.net/relu1-and-relu6-activation-functions-with-benefits-and-drawbacks/.

[46] J. P. a. A. Gibson, " Deep Learning," *O'Reilly Media,* 2014.

[47] A. Géron, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow," *O'Reilly Media,* 2019.

[48] N. Buduma, "Fundamentals of Deep Learning," *O'Reilly Media,* 2017.

[49] "Demystifying Hyperparameters in Machine Learning Models," May 31, 2023.

[50] J. B. Y. L. A. S. a. P. V. M. M. Bronstein, " Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine,* p. 18–42, July 2017.

[51] "Graph Neural Networks — deep learning for molecules & materials," Jun. 01, 2023.

[52] "Deep Learning: Unleashing the Power of Artificial Intelligence," May 18, 2023.

[53] G. C. C. R. L. B. Petar Velickovi, "GRAPH ATTENTION NETWORKS," in *ICLR 2018*, 4 Feb 2018.