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

VIA

Par :

BAGHDAD Amine

Sur le thème

Building an Innovative E-Commerce Platform Powered by Session-Based Recommendations

Soutenu publiquement le 19 / 06 / 2025 à Tiaret devant le jury composé de :

Mr. TALBI Omar	MCA	Université de Tiaret	Président
Mr. BOUDAA Boudjemaa	MCA	Université de Tiaret	Encadrant
Mr. BERBER EL-Mehdi	MAA	Université de Tiaret	Examinateur
Mr. DAOUD Mohamed Amine	MCB	Université de Tiaret	Représentant de l'incubateur
Mr. CHAHBI Tayeb	Direction du commerce de Tiaret		Représentant du partenaire économique

2024-2025

Acknowledgements

First and foremost, I am deeply grateful to Allah, the Most Gracious and the Most Merciful, for granting me the strength, patience, and opportunity to undertake and complete this thesis. I would like to express my heartfelt appreciation to my family, especially my parents, whose unwavering love, support, and encouragement have been the foundation of my academic journey. My sincere thanks also go to my brothers, sisters, nephews, nieces, and all my close relatives and friends, whose presence and kind words have provided constant motivation and comfort throughout this challenging yet rewarding experience. I am profoundly thankful to Professor BOUDAA Boudjemaa for his invaluable supervision, insightful feedback, and continuous encouragement throughout the course of this research. I would also like to extend my sincere gratitude to PhD student DJENANE Mouloud Amine for his helpful advice and guidance. Finally, I would like to express my gratitude to the members of the jury for their time, thoughtful evaluation, and constructive comments on my thesis.

Baghdad Amine

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
RS	Recommender System
SBRS	Session-Based Recommendation Systems
ANN	Artificial Neural Networks
RNN	Recurrent Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
CF	Collaborative Filtering
CBF	Content-Based Filtering
GDPR	General Data Protection Regulation
CCPA	California Consumer Privacy Act
BPTT	Backpropagation Through Time

Abstract

Businesses and consumers' online shopping habits have been significantly altered by the rise of e-commerce in recent years. To assist users in finding products and streamlining the shopping experience, major platforms such as Amazon, eBay, and Alibaba employ recommendation algorithms. These algorithms are key to enhancing user satisfaction and driving sales. Although content-based approaches and collaborative filtering are popular traditional recommendation systems, they struggle to adapt to the rapidly shifting preferences of users. This is where Session-Based Recommendation Systems (SBRSs) come in. Without relying on long-term user data, SBRSs focus on predicting the next product a user will engage with based solely on their actions during the current session. This strategy is especially helpful on e-commerce platforms, where users often browse anonymously and make quick decisions. The purpose of this thesis is to use long short-term memory (LSTM) networks to enhance session-based recommendations in e-commerce. Due to their capacity to recognize patterns in user behavior over time, LSTM networks are ideal for this task. By predicting the next item a user will interact with based on session activity, these models significantly improve recommendation quality. To demonstrate their practical impact, an e-commerce platform called VIA was developed, showcasing how LSTM-based session-based recommendation models can deliver highly personalized shopping experiences.

Keywords: Recommender system, Session-based recommendation system, Recurrent neural networks, LSTM, E-commerce platform, VIA.

General Introduction

1 Background

E-commerce has witnessed remarkable growth over the past decade, driven by the digital revolution, the widespread use of smartphones, and expanded internet access. This transformation has radically changed consumer behavior, making online shopping a common daily activity and forcing digital platforms to compete to provide the best possible user experience to increase engagement and sales.

In this context, recommendation systems are becoming an essential tool that assists platforms in providing consumers with tailored content and products based on their preferences and actions. These systems have evolved from traditional techniques such as collaborative filtering, which relies on similar user behavior, content-based methods that focus on the characteristics of user favorites, and hybrid models that combine the two.

The first implementation of the recommendation system (RS) concept emerged in 1979 in the form of *Grundy*, a computer-based library that suggested books to its users. The Tapestry, the first commercial RS, was launched in the early 1990s, in response to people's struggles to access and pick relevant data from the internet [1]. Although these systems work well, their efficacy is limited when interacting with new or unknown users because they usually rely on past user data, such as browsing history or ratings.

With the increasing number of unregistered visitors to e-commerce platforms, a new challenge has emerged—how to deliver relevant recommendations without any prior user data. Due to this difficulty, a novel class of recommendation systems called session-based recommendation systems (SBRS) has emerged, which focus on analyzing interactions that occur only during the current session.

The evolution of session-based recommender systems can be divided into two main phases: the model-free phase from the late 1990s to the early 2010s, and the model-based phase from the early 2010s onwards. The first phase drew heavily from data mining methods like pattern mining, association rule mining, and sequence mining. During this time, researchers mostly aimed to uncover frequent or sequential patterns and rules from user sessions to power recommendations. This approach peaked around the mid-2000s, leading to numerous pattern-based, rule-based, and sequence-based techniques.

The second stage started with the advancement of machine learning and statistics methods, especially those that worked with time-series data. This phase was characterized by the introduction of models like Markov chains and recurrent neural networks (RNNs), which enabled more complex modeling of sequential relationships in user behavior. Gated Recurrent Units (GRUs), one type of RNN, have been shown to be good at capturing sequential patterns, particularly in sparse recommendation data. The GRU4Rec model by Hidasi et al. (2016) [2] is a notable addition from this phase, as it applies GRUs particularly to session-based recommendation tasks and performs well in predicting user actions inside a session.

Since the rapid advancement of deep learning technologies around 2017, model-based recommender systems have gained even more attention, leading to a wave of research into neural architectures for next-item and next-basket prediction. The ability of RNNs to model complex, time-dependent patterns in user behavior has made them particularly useful for session-based recommendation systems, where the challenge is to predict the next action or item a user will interact with, based on a limited set of interactions within the current session.

2 Problem Statement

In many online platforms, particularly in e-commerce, users often browse anonymously, which limits the availability of long-term user data. Session-based recommendation systems (SBRS) aim to address this by focusing on a user's current session. However, traditional SBRS approaches use basic models or fixed rules that struggle to capture the sequential and dynamic nature of user behavior.

To address this challenge, this thesis explores the use of Long Short-Term Memory (LSTM) networks—a kind of Recurrent Neural Network (RNN) that is well-known for its capacity to represent temporal patterns and long-range dependencies in sequential data. By leveraging LSTM's ability to retain contextual information throughout item interactions, this work aims to improve the accuracy and relevance of session-based recommendations, driven by the following question:

> *"How can LSTM-based models be effectively applied to capture the sequential and evolving patterns of user behavior in session-based recommendation systems to enhance recommendation quality in e-commerce platforms?"*

3 Delimitation

This thesis focuses on improving next-item prediction in session-based recommendation systems by analyzing item-item and item-session relationships within anonymous browsing sessions. Using only session-level data (item IDs and timestamps), we examine how LSTM networks can effectively model sequential patterns without incorporating user profiles, long-term history, or external contextual factors (e.g., demographics, device type).

The study is evaluated on two public datasets and is strictly limited to the e-commerce domain. It purposefully focuses only on next-item prediction, rather than broader recommendation tasks such as session clustering or next-basket prediction.

In addition to providing a targeted baseline for future extensions, this research aims to deepen our understanding of pure session-based recommendation systems by concentrating on LSTM networks' capacity to capture sequential dependencies.

4 Approach

To tackle the task of session-based recommendation, a model based on Long Short-Term Memory (LSTM) networks is presented. The proposed approach treats user sessions as sequences of item interactions, ordered by timestamp and represented using mapped item indices. Only session-level data is used—no user profiles or external context.

To learn the temporal dependencies between items, the embedded session sequences are passed through an LSTM network. The model predicts the next item by leveraging the patterns captured across previous interactions.

The system is evaluated on two public e-commerce datasets using standard top-k metrics such as Hit Rate and MRR, with a specific focus on next-item prediction in anonymous browsing sessions.

5 Outline

After this general introduction, the thesis is organized into **four chapters**, each detailing a key component of the research, in addition to a general conclusion. The structure is as follows:

- **CHAPTER ONE: SESSION-BASED RECOMMENDER SYSTEMS**
  Introducing recommender systems with a special focus on session-based recommender systems, exploring their relevance and application in e-commerce. It provides foundational knowledge for understanding how these systems function and highlights key challenges and opportunities in session-based recommendations.

- **CHAPTER TWO: RECURRENT NEURAL NETWORKS (RNNs)**
  exploring the foundational concepts of Artificial Intelligence, Machine Learning, and Deep Learning, with A special focus on RNNs. It explores the history and development of RNNs, their architecture, and their ability to model sequential dependencies in user behavior. The section highlights key variants of RNNs, such as LSTM networks and GRUs, and their applications in session-based recommendation systems, where they capture temporal patterns to improve recommendation accuracy.

- **CHAPTER THREE: A SESSION-BASED RECOMMENDER SYSTEM USING LSTM FOR E-COMMERCE**
  Presenting the practical implementation of a session-based recommender system using Long Short-Term Memory (LSTM) networks. It covers data preprocessing, model architecture, training, and evaluation. Results are analyzed and discussed, highlighting the strengths, limitations, and possible improvements of the proposed approach.

- **CHAPTER FOUR: BUILDING AN E-COMMERCE PLATFORM POWERED BY SESSION-BASED RECOMMENDATIONS**
  Exploring the design and development of an e-commerce platform integrated with a session-based recommender system. It discusses the platform's architecture, design considerations, and the challenges of incorporating real-time personalized recommendations into a functional web application.

- **GENERAL CONCLUSION**
  Summarizing the main findings of the thesis, emphasizing the potential of LSTM-based session recommender systems in e-commerce. It also outlines key limitations and future research directions.

Chapter 1

Session-Based Recommender Systems in E-Commerce

1.1 Introduction

In today's digital world, where consumers are constantly exposed to an overwhelming number of choices, recommender systems have emerged as indispensable resources for assisting them find relevant products and content. These systems are designed to facilitate the user experience by providing recommendations based on personal preferences and requirements, which will ultimately enhance decision-making and user satisfaction.

Conventional recommender systems, which rely on long-term user data like browsing and purchase history, have worked well in a variety of scenarios. However, they often struggle to adapt to the dynamic and short-term nature of user preferences. With the increasing use of mobile devices, private browsing, and stricter privacy regulations, gathering long-term behavioral data has become more difficult. As a result, strategies that emphasize real-time user behavior over persistent user identities are becoming more and more necessary.

Session-based recommender systems provide a promising solution to this problem. Instead of depending on past user profiles, they focus on the actions taken within a single session — capturing the user's current intent based on their immediate behavior. This approach has drawn a lot of interest in recent years, especially in environments where users remain anonymous or historical data is unavailable.

This chapter delves deeply into the concept of session-based recommendation. We'll examine the motivation behind its development, the methods that contribute to its efficacy, and its benefits and drawbacks compared to traditional approaches. Through this, we aim to provide a clear understanding of how session-based systems are shaping the future of personalized recommendations in e-commerce and beyond.

1.2 The Role of Recommender Systems

Recommender systems are AI-driven algorithms, usually associated with machine learning, that leverage large-scale user data to suggest additional products or services to consumers [3]. These systems are now essential parts of many digital platforms, such as online news platforms, streaming services like Netflix and Spotify, and e-commerce websites like Amazon. In today's data-rich digital landscape, users are often overwhelmed by the vast amount of available content. By prioritizing and filtering information, recommender systems help users make quicker and better-informed decisions. They greatly improve user satisfaction, engagement, and overall experience by providing tailored recommendations based on user interactions and preferences.

1.2.1 Evolution and History

The development of recommender systems dates back to the early 1990s, closely tied to the rise of the internet and the need to help users filter growing volumes of online content [4]. One of the earliest influential systems was GroupLens, introduced in 1994, which used collaborative filtering to recommend Usenet news articles and inspired the establishment of a dedicated research group at the University of Minnesota [5]. Around the same time, systems like Ringo, designed for music recommendations [6], and the Video Recommender, built for multimedia content [7], demonstrated how such technologies could be adapted to various domains.

By 1996, the commercial potential of recommendation engines became evident with the founding of Net Perceptions, a company that offered marketing-focused recommender engines to major retailers like Amazon and Best Buy. These early developments bridged academic research and real-world applications, setting the stage for broader adoption.

The field experienced significant momentum during the Netflix Prize competition (2006–2009), which encouraged global researchers to improve recommendation algorithms using large-scale user-item interaction data. Matrix factorization techniques gained prominence during this era, marking a shift from traditional collaborative filtering models to more scalable and accurate approaches [8]. Recognizing the growing importance of this area, the ACM Recommender Systems Conference (RecSys) was launched in 2007 and has since become the leading annual conference in the field [9].

Since the late 2010s, deep learning has driven a new wave of innovation in recommender systems. Several industrial-scale models have been introduced to improve the performance of recommendation engines, including Wide Deep [10], DeepFM [11], and YouTubeDNN [12]. Benchmarking efforts like FuxiCTR [13] and broader reproducibility initiatives outlined in [4] have further pushed the field toward rigorous and standardized evaluation practices. Research has also expanded into areas such as fairness, privacy, and causal inference, reflecting the complex social dimensions of recommendation in real-world platforms.

As noted in the Brief History of Recommender Systems [4], this period marked a shift from focusing solely on accuracy metrics to considering user experience and personalization quality — a trend that continues to shape the design of modern recommender systems.

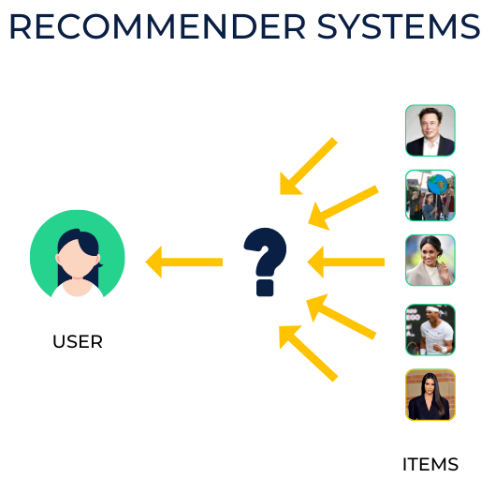


Figure 1.1: Example of recommendation system

1.2.2 Traditional Techniques

There is an extensive number of recommender system models, but they are generally divided into three main categories: collaborative filtering, content-based filtering, and hybrid methods. These traditional approaches have served as the foundation for personalized recommendation systems across domains such as e-commerce, media streaming, and social platforms. Despite the growing use of more advanced techniques like deep learning, these methods remain widely adopted due to their simplicity and effectiveness.

Collaborative Filtering

Collaborative filtering (CF) is the process of filtering or evaluating items through the opinions of other people [14]. In other words, rather than focusing on product features, CF draws patterns from user behaviors and shared preferences. The core idea is simple: users who liked similar items in the past are likely to enjoy similar items in the future. By analyzing patterns in user-item interactions, the system identifies groups of users with similar tastes or items that tend to be liked together, and makes predictions accordingly. This technique is widely used in domains such as e-commerce, streaming services, and social platforms to provide personalized suggestions(see Figure 1.2)[1].

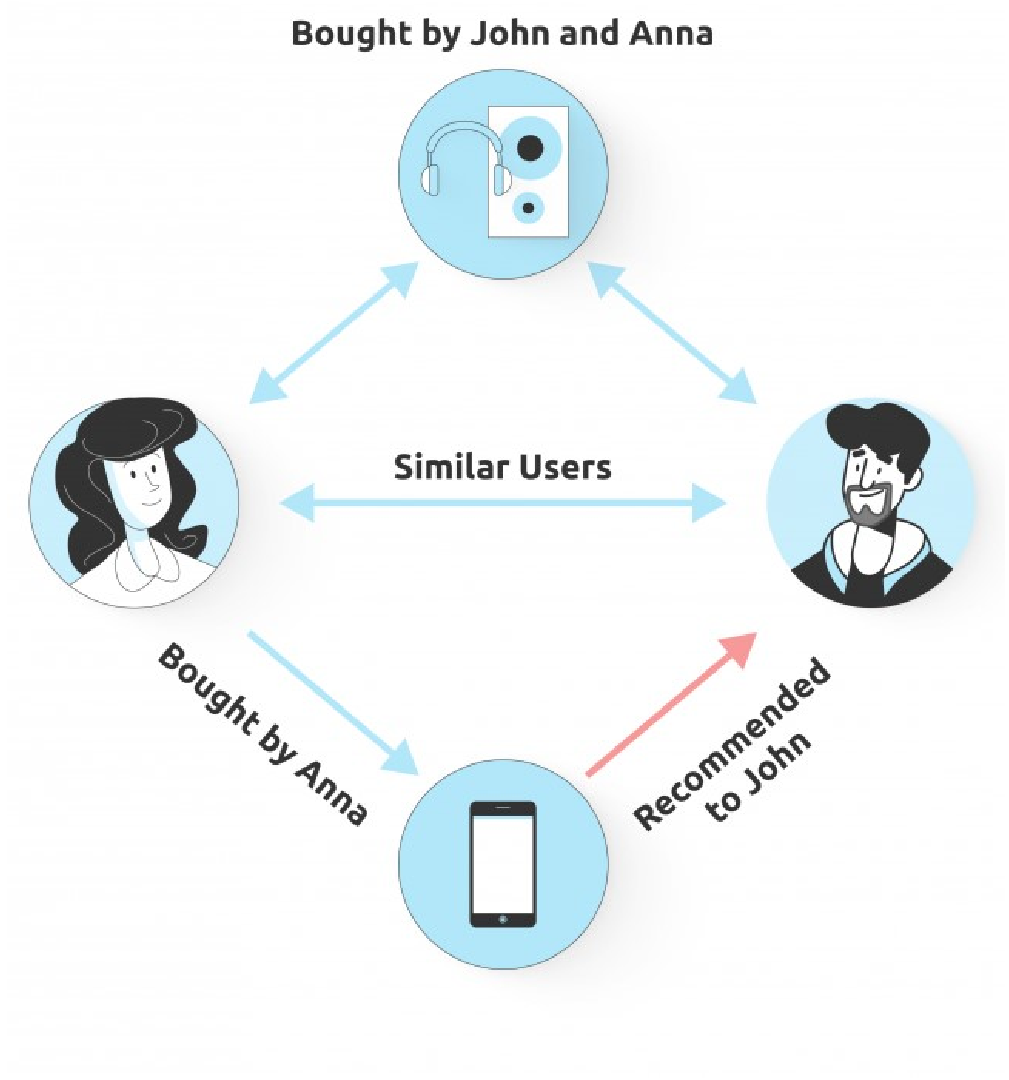


Figure 1.2: Example of collaborative filtering

Content-Based Filtering

Content-Based Filtering (CBF) recommends items to users by analyzing the features of products they've shown interest in—such as items they've viewed, liked, or purchased. In simpler terms, it tries to match the features of new items with those of items the user already likes. For example, in an e-commerce platform, if a user frequently browses fitness trackers and headphones, the system may suggest other wearable tech products with similar features(see Figure 1.3)[1]. As noted by Zhou et al., content-based methods "primarily offer recommendations by comparing item features with users' historical preferences" [15]. Similarly, Roy and Dutta explain that items rated positively are compiled into a user profile, and then similar items are recommended to that user [16].

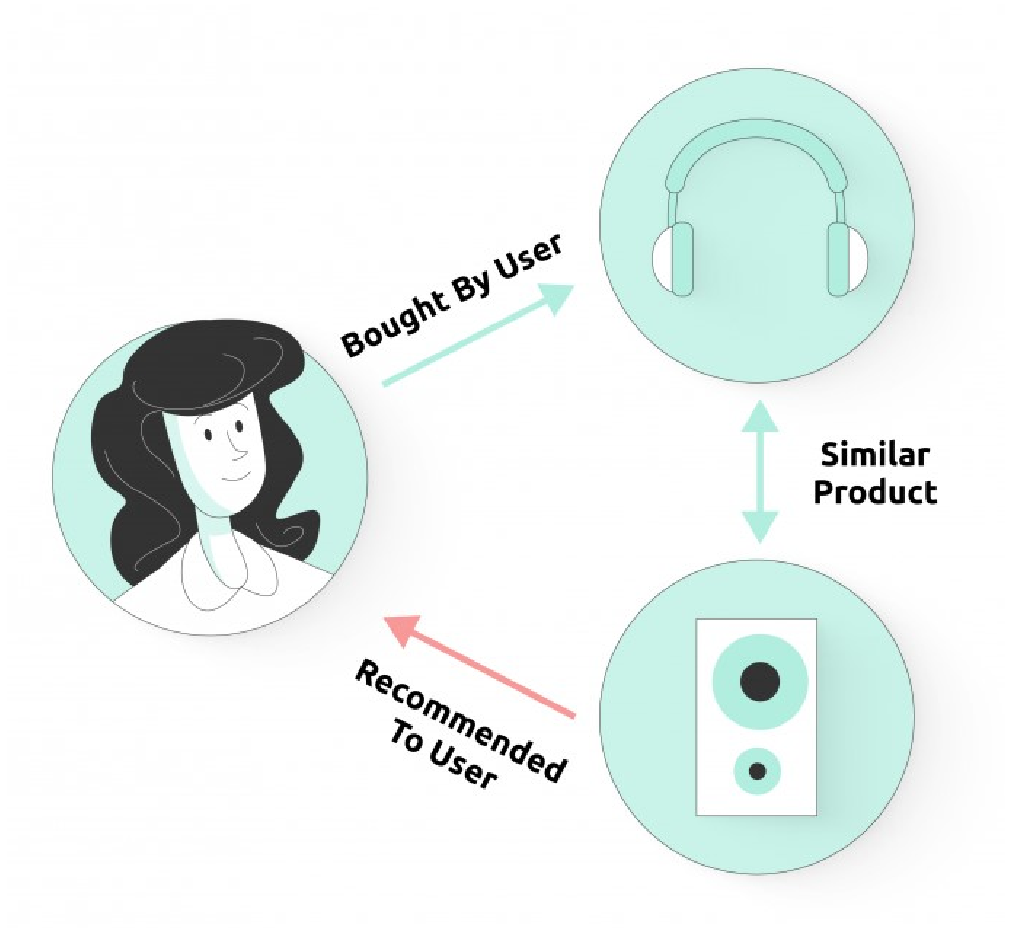


Figure 1.3: Example of content-based filtering

Hybrid Models

Hybrid recommender systems are recommendation engines that combine two or more different recommendation techniques—often collaborative filtering and content-based filtering—in order to improve suggestion quality(see Figure 1.4). These systems are designed to overcome the limitations of individual methods by leveraging their complementary strengths. For example, collaborative filtering often has trouble making suggestions for brand-new users or items because it relies on past interactions—this is known as the cold-start problem. Meanwhile, content-based filtering can become too limited by recommending only items similar to what the user already likes, making suggestions feel repetitive over time.

To address these challenges, hybrid systems integrate both approaches. By combining collaborative and content-based methods, these models can better handle challenges while improving the ability to offer more accurate and diverse recommendations. Instead of relying on just one method, hybrid systems use a mix of strategies—such as giving weight to different recommendation sources, switching between methods based on the situation, or enriching the data used for predictions. This kind of integration allows the system to consider both what users like and the features of the items themselves, which often leads to more accurate, diverse, and personalized recommendations across different applications [17][18].

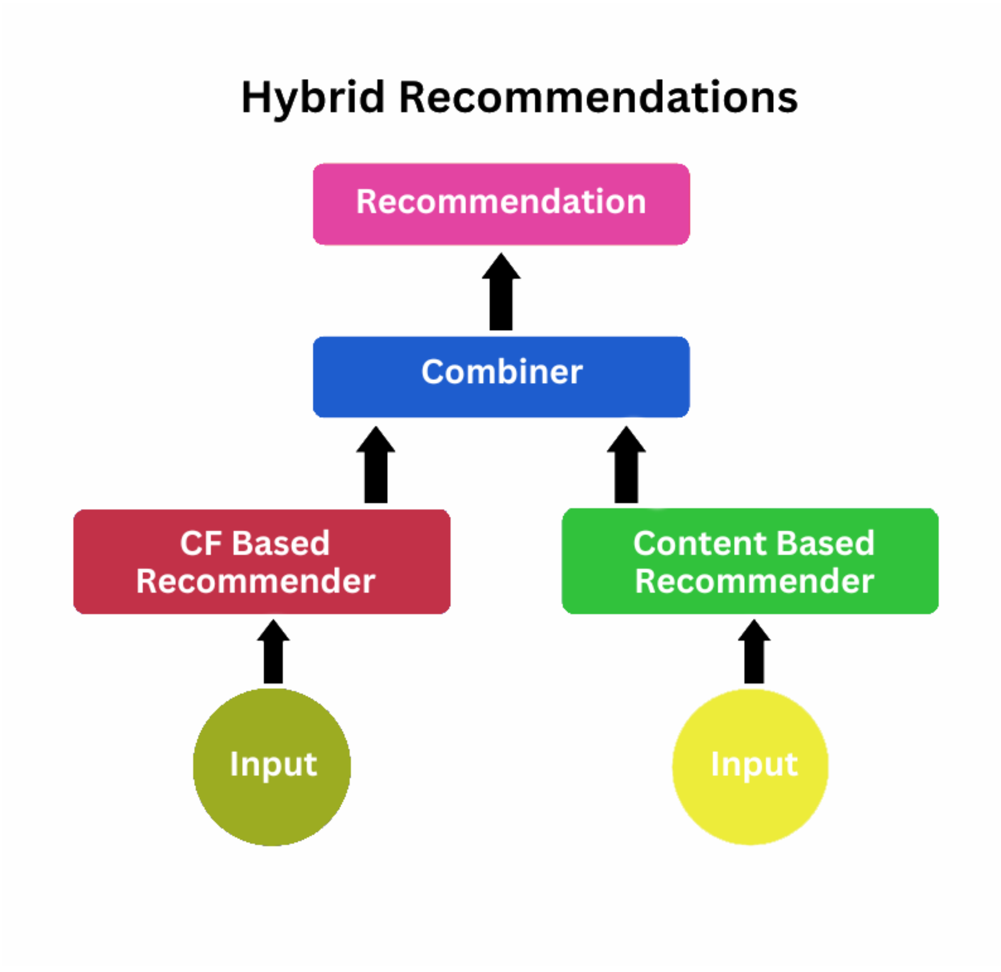


Figure 1.4: Example of Hybrid Models

1.2.3 Challenges and Limitations

Although traditional recommender systems—such as collaborative filtering (CF), content-based filtering (CBF), and hybrid approaches—have achieved notable success across industries, they still face fundamental challenges. These limitations restrict their performance, scalability, and trustworthiness in real-world settings. The following summarizes the most widely recognized issues in the literature and practice:

- **Cold Start Problem:** When either users or items are new to the system, there is insufficient data to generate reliable recommendations. This is divided into:

  - User cold start, where no preference history exists for new users.
  - Item cold start, where newly introduced items lack interactions.

  Addressing cold start remains a core challenge in both CF and CBF methods [19].

- **Data Sparsity:** In most systems, users interact with only a small portion of the available items, resulting in sparse interaction matrices. This undermines the effectiveness of similarity-based approaches like CF [20].

- **Scalability:** As the volume of users and items increases, so does the computational complexity. Simple memory-based CF methods often cannot handle large-scale applications efficiently without optimization or distributed computing [21].

- **Preference Drift:** User interests change over time, yet many models fail to adapt dynamically. Ignoring temporal dynamics can lead to stale or irrelevant suggestions [22].

- **Synonymy:** Different names or terms referring to the same item (e.g., "cellphone" vs. "smartphone") may be treated as unrelated, reducing recommendation quality in CBF methods that lack semantic understanding [20].

- **Shilling Attacks:** Attackers may inject fake ratings to manipulate system output—either boosting or downgrading specific items. CF systems are particularly vulnerable to such adversarial behaviors [23].

- **Limited Novelty and Diversity:** Many RSs over-recommend popular or highly similar content, reinforcing a "filter bubble" and failing to surprise or engage users with novel items [24].

- **Privacy Concerns:** RSs rely on collecting sensitive user data such as browsing behavior, location, or ratings. This raises ethical and legal challenges [25].

- **Dependence on Rich Metadata:** CBF and hybrid approaches often rely on detailed, structured metadata (e.g., genres, descriptions). When this data is missing or incomplete, recommendations degrade significantly [20].

1.3 Session-Based Recommendations

Session-Based Recommendation Systems (SBRS) have become an essential tool when the system cannot rely on user identification or the establishment of permanent profiles. This is often the case in online platforms such as e-commerce, where a large number of users browse anonymously or do not return frequently. Compared to traditional recommenders that rely on a user's history across multiple sessions, SBRS concentrate on learning from the short-term behavior that occurs within a single session [2].

The main goal of an SBRS is to predict the next item a user might interact with based on their actions during the current visit. To determine immediate intent, these systems analyze the sequence and type of user interactions over time t, such as product views or link clicks.

Various techniques are used for this purpose, ranging from simple rule-based or Markov models to advanced deep learning approaches like Recurrent Neural Networks (RNNs) and Graph Neural Networks (GNNs) [26].

1.3.1 Defining the Paradigm

The session-based recommendation paradigm focuses on making predictions from a user's interactions within a single visit, rather than relying on a long-term profile built over many visits. A "session" is defined as a short sequence of events—views, clicks, add-to-cart actions—separated by an inactivity timeout (commonly around 30 minutes) [2](see Figure 1.5). This approach is especially useful on e-commerce sites where many users browse anonymously or infrequently, making persistent user tracking challenging.

Given the limited data within a session, these systems must infer user intent from minimal interaction. Recurrent Neural Networks (RNNs) are widely used in this context because they can process sequences of arbitrary length and maintain an internal state that captures how earlier actions influence later ones. In addition to the order of events, the type of action (e.g., view vs. add-to-cart) provides extra clues about user interest; encoding these action types alongside item identifiers helps the model distinguish casual browsing from strong purchase intent [26].

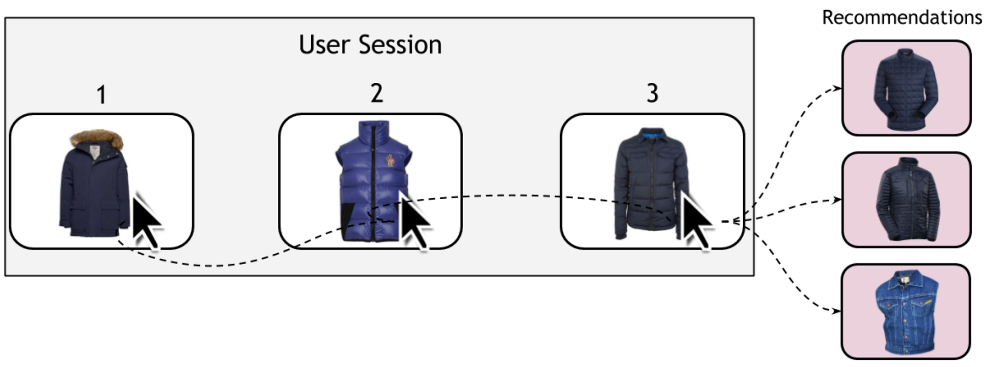


Figure 1.5: An example of session-based recommendation

1.3.2 Capturing User Interaction Behavior

In session-based recommendation, the system's understanding of what a user wants comes entirely from the actions they take during a single visit. To build this understanding, platforms such as e-commerce sites log several main types of events:

1. **Page views:** Every time a user loads or reloads a product or category page, the system notes which item or page was displayed. This basic signal shows what caught the user's eye initially.

2. **Clicks:** When a user clicks on an item—whether an image, title, or "details" link—it indicates deeper interest than a mere view, suggesting the user wants more information.

3. **Add-to-cart actions:** Placing an item into the shopping cart is a strong signal of purchase intent. It tells the system the user is seriously considering buying that product.

4. **Purchases:** The final conversion event confirms the user's decision and provides ground-truth data about what users ultimately want.

5. **Micro-behaviors:** Beyond these coarse events, many systems also capture finer signals—such as time spent on a page, scrolling depth, mouse hovers, or image-zoom interactions—to distinguish casual browsing from genuine interest [27].

Each event type carries a different strength of intent, and understanding this helps the recommender prioritize which signals to trust. For example, a series of views without any clicks frequently denotes casual exploration, whereas a view followed by a click and an add-to-cart action indicates increasing interest. Purchases are the strongest training signals and provide official confirmation of intent. Micro-behaviors fill in the gaps by indicating hesitation or deeper consideration—long pauses or repeated zooms suggest the user is evaluating details closely. By combining event sequences, action weights (e.g., purchase $>$ add-to-cart $>$ click $>$ view), and micro-behavior cues, a session-based model can develop a detailed, real-time picture of what the user is likely to do next [28].

1.3.3 Data Collection and Session Formation

In session-based recommendation, the quality of the input data largely determines how well the model can learn and predict user behavior. Since these systems typically operate without long-term user histories, they rely heavily on accurately formed sessions—structured sequences of interactions that capture a user's short-term goals during a single visit.

Capturing User Events: User interactions are first recorded through event logs generated by web or app activity. These logs typically include timestamps, event types (e.g., view, click, purchase), product or content IDs, and user identifiers (when available). Each row represents a discrete event and serves as a building block for later session construction. Platforms such as YouTube, Amazon, and Netflix rely on this type of clickstream data to feed their session-based models [12, 2].

Session Identification: Identifying sessions from raw interaction logs can be challenging, especially when users are anonymous or not logged in. In these cases, sessions are inferred using common techniques:

1. **Time-Based Segmentation:** One of the easiest ways to split user activity into sessions is by looking at breaks in their actions. If a user stops interacting for a while, it usually indicates that a new session has started. This method is widely used in web analytics and session-based recommendation systems. The assumption is that when someone takes a long break, they're likely no longer focused on the same task. However, some researchers argue that using a fixed time threshold may not generalize across users, prompting newer methods that adapt based on individual behavior [29].

2. **Explicit Session Tracking:** When available, session identifiers (such as cookies or session tokens) offer the most accurate way to delineate sessions. These identifiers are typically generated by the server or embedded in the client (e.g., browser cookies) and allow for precise session boundaries [30].

3. **IP and User-Agent Grouping:** In the absence of unique user IDs or session tokens, a combination of IP address and user-agent string can serve as a proxy to group interactions from the same user within a time window. This method is more error-prone due to dynamic IPs, shared devices, or similar user-agent strings, but can still be useful in anonymous settings [31].

Structuring Sessions for Analysis: Once user sessions have been identified, they need to be organized into a format suitable for analysis. In session-based recommendation, a session is typically represented as a chronologically ordered sequence of user interactions. Each session captures the short-term behavior of a user during a single visit or interaction period.

The most common way to structure a session is as a list of item identifiers, where each item reflects an interaction (such as viewing a product or clicking a link). These sequences preserve the order of events, which helps reveal patterns in user behavior, such as repeated actions or trends in interest over time.

In addition to item IDs, sessions can include optional metadata:

- Timestamps that mark when each interaction occurred,
- Action types (e.g., view, click, cart addition),
- Item categories or brands for contextual understanding,
- Session length or position indicators to describe the interaction's place within the session [32].

How much detail is included in the session representation depends on the use case and the available data. The goal is to preserve enough meaningful information about user behavior without introducing noise.

1.3.4 Privacy and Security

In session-based recommendation, systems process streams of user interactions like views, clicks, and cart actions to determine short-term intent. Even though these models don't build long-term user profiles, they still handle data that can reveal personal preferences. Accordingly, privacy and policy considerations focus on how session data is collected, stored, and protected.

Data Minimization Anonymity: Platforms in session-based recommendation record only the user actions needed to generate accurate short-term suggestions such as page views, clicks, and purchases, and remove any details that could directly identify someone. Instead of storing a permanent user ID, systems often issue a short-lived token for each visit, so it's hard to link one session to the next. This "collect only what's necessary" idea comes from the GDPR's data-minimization rule, which says you must limit personal data to what's needed for your purpose [33].

Consent Transparency: Users need to know when their actions are being tracked and why. Session-based systems typically display a simple notice—often as a cookie banner or privacy pop-up—that explains which interaction data (page views, clicks, cart adds) will be collected and how it will be used to improve recommendations. Consent mechanisms let users opt in or out of non-essential tracking, ensuring that only agreed-upon data enters the session logs. Transparency also means providing a clear, concise privacy notice that describes what session data is stored, how long it's kept, and how users can request its deletion [34].

Regulatory Compliance: Session-based recommenders must follow the rules set by major privacy laws, even though they don't build long-term profiles:

- **GDPR (EU):** Users have the right to know what session data is collected, to access it, and to request its deletion or correction. Any tracking (including cookies or tokens) requires a lawful basis—often user consent—and platforms must document how session logs are handled [35].
- **CCPA (California):** California residents can ask what personal information (including session events) has been gathered, opt out of its sale, and demand that companies delete that data [36].

These regulations ensure that, even for anonymous sessions, users retain control over their interaction data and can enforce data-deletion or access rights.

Data Security: Even though session data is short-lived, it must be protected against interception and tampering. The most critical measure is to encrypt all session traffic in transit using up-to-date TLS protocols (currently TLS 1.3), which prevents eavesdroppers from reading or modifying interaction logs as they travel between client and server [37].

1.4 Session-Based Recommendation in E-commerce

In e-commerce, turning brief, anonymous visits into sales is a major challenge. Session-based recommendation systems (SBRS) tackle this by relying solely on interactions from the current session—no long-term user data required. By analyzing real-time behavior, they deliver timely, relevant product suggestions that boost the chances of conversion. This approach is especially effective for engaging first-time or unidentified users with personalized recommendations.

1.4.1 Business Drivers

The growing adoption of Session-Based Recommendation Systems (SBRS) in e-commerce is driven by several strategic needs:

- **Addressing Dynamic User Preferences:** SBRS can cater to the short-term and evolving preferences of users by generating relevant suggestions based solely on interactions within an ongoing session, unlike traditional methods requiring extensive user history [2].

- **Serving Anonymous/New Users:** They are particularly beneficial for users who are anonymous, use private browsing, or are new visitors with no historical data, as recommendations rely on individual browsing behavior within a single session [2].
- **Adapting to Evolving Consumer Behavior:** With increased mobile usage and privacy concerns making long-term data collection difficult, SBRS offer a solution by focusing on immediate user intent captured in a single session.
- **Real-time Personalization:** This focus enables real-time personalization of the user experience, which can enhance engagement and potentially increase conversion rates even without detailed user profiles.

1.4.2 Use Cases in Online Retail

Session-based recommendation systems (SBRS) offer numerous practical and high-value applications in online retail, adapting in real-time to user intentions to enhance customer experience and boost sales. Key use cases include:

- **Homepage Personalization for New/Unauthenticated Visitors:** Analyzing initial interactions (e.g., clicks, viewed products, search terms) to dynamically display relevant promotional banners, personalized reminders, trending products, ensuring a more engaging and relevant browsing experience from the beginning [38].
- **Product Page Recommendations (Next Item Prediction):** When a user views an item, SBRS can suggest:
  - Products "frequently viewed together."
  - Items that "customers who viewed this item also viewed."
  - Alternative products based on current item attributes and session activity.

  This form of next item prediction aids product discovery and can increase average order value by suggesting complementary items or upgrades [38].
- **Personalization of Internal Search Results:** Refining search result order based on items viewed or added to the cart during the current session, weighting results to reflect contextual preferences like brand or price range interest.
- **Shopping Cart Recommendations (Next Basket Prediction):** Just before order finalization, SBRS can suggest:
  - Items frequently purchased with those already in the cart (e.g., accessories).
  - Forgotten items that similar users purchased.

  This application, a form of next basket prediction or basket completion, aims to increase order value and remind users of relevant items. Displaying previously viewed products can also streamline the purchase process [39].

- **Real-Time Abandoned Cart Recovery:** Displaying targeted recommendations in exit-intent pop-ups or banners if a user attempts to leave with items in their cart, encouraging purchase completion or exploration of alternatives based on session context.

These applications highlight SBRS's flexibility in integrating across the online customer journey, offering dynamic personalization without relying on extensive historical data, a key advantage in modern e-commerce.

1.4.3 Limitations and Considerations

Despite their advantages, SBRS present several challenges that must be considered:

- **Data Volume and Quality:** A significant hurdle is the often limited volume and quality of data available within a single session. Short or low-interaction sessions can make it difficult to accurately infer user intentions, leading to less relevant recommendations [40].
- **Privacy and Data Protection:** Collecting and analyzing user behavior, even ephemeral session data, raises ethical and regulatory concerns. Compliance with regulations like GDPR and CCPA is crucial, requiring measures such as encryption, anonymization, and transparency regarding data use. Recommendation engines' reliance on personal data can lead to concerns about privacy, consent, security, data ownership, and ethical issues like bias and manipulation [41].
- **Algorithmic Complexity:** Developing models to accurately infer intentions from often short, noisy action sequences demands sophisticated approaches (e.g., deep learning), which can be costly to develop and maintain.
- **Cold Start Problem:** While less severe than in history-based systems, the cold start issue can still affect SBRS in sessions with very few initial interactions, making it challenging to provide meaningful recommendations from the very first user actions.

1.5 Conclusion

Session-based recommendation systems have become a powerful alternative to traditional recommender approaches, especially in contexts where long-term user data is unavailable or unreliable. SBRS provides a dynamic and responsive method of determining user intent by concentrating on short-term behavioral patterns within a single session, allowing for rapid and customized recommendations.

While these systems bring considerable advantages including the ability to support anonymous users, enabling real-time personalization, and adapting to changing consumer behaviors, they also face challenges related to data sparsity, algorithmic complexity, and ethical considerations. Understanding these trade-offs is essential for designing effective SBRS solutions.

Looking ahead, future advancements will likely focus on overcoming current limitations while expanding the scope and accuracy of session-aware personalization across industries.

The next chapter will explore how Recurrent Neural Networks (RNNs) are particularly well-suited to capture the sequential nature of session data, thereby enhancing the effectiveness of session-based recommendation systems.

Chapter 2

Recurrent Neural Networks

2.1 Introduction

Many tasks in modern computing involve sequential data, such as text, speech, time series, or user interaction sessions, where the order of inputs plays a critical role. To successfully handle such data, models must be able to capture dependencies between events over time.

Recurrent Neural Networks (RNNs) are designed specifically for this use. They are now necessary tools for many sequence-based tasks, such as e-commerce session-based recommendation systems.

This chapter provides an introduction to the key concepts needed to understand RNNs. It starts with a brief overview of artificial intelligence, machine learning, and deep learning, and then focuses on neural networks and their architecture. Finally, it explores RNNs and their main variants, setting the foundation for the recommendation model used in this thesis.

2.2 Artificial Intelligence

Artificial intelligence (AI) is the field of computer science concerned with creating machines and software capable of performing tasks that normally require human intelligence—such as understanding language, recognizing patterns, solving problems, and learning from experience. It combines techniques from mathematics, statistics, and engineering to design algorithms that can process data, adapt to new inputs, and improve performance over time. Common examples include virtual assistants that interpret speech, recommendation systems that tailor content, and autonomous vehicles that navigate complex environments.

The foundational vision for AI was laid out in the 1955 Dartmouth proposal, which conjectured that "every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it" [42].

2.3 Machine Learning

Machine learning is a subfield of artificial intelligence where computer programs automatically improve their performance on a given task through experience, rather than relying on explicit programming. In other words, these programs use data to adjust internal parameters so that, over time, they make more accurate predictions or decisions. A concise operational definition is:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." [43]

2.3.1 Types of Machine Learning

Machine learning (ML) algorithms can be classified into four main categories—supervised, unsupervised, semi-supervised, and reinforcement learning—based on how they use labeled and unlabeled data and how they learn from experience [44]. Each approach differs in how it processes data and builds models, adjusting internal parameters during training to make accurate predictions or decisions on new data [44]. Core concepts in ML include features (input variables), labels (desired outputs), model fitting (learning parameters), generalization (performance on unseen data), and the balance between overfitting and underfitting [45].

Supervised Learning

Algorithms learn from a dataset of input–output pairs, finding a function that maps inputs to known labels. During training they minimize a loss function measuring prediction errors. Common tasks are classification (e.g., spam vs. not-spam) and regression (e.g., predicting house prices) [45].

Unsupervised Learning

Algorithms receive only inputs and must discover hidden structure—such as clusters or lower-dimensional representations—without guidance from labels. Typical methods include K-means clustering and principal component analysis (PCA) for dimensionality reduction [45].

Semi-Supervised Learning

This blends a small amount of labeled data with a large pool of unlabeled data. An initial model trained on labeled examples then pseudo-labels the unlabeled set, and retraining on the combined data improves accuracy when labels are scarce or costly [46].

Reinforcement Learning

An agent interacts with an environment by taking actions and receiving rewards or penalties. The goal is to learn a policy that maximizes cumulative reward over time, balancing exploration of new actions with exploitation of known good ones. Classic examples include game-playing AIs and robotic control systems [44].

2.3.2 Limitations and Challenges

While machine learning has achieved remarkable progress, it still faces key challenges that limit its effectiveness .

- **Dependence on Data Quality:** Machine learning models require large volumes of high-quality data. Incomplete, biased, or noisy data can lead to inaccurate predictions and poor model performance [47].
- **Lack of Transparency and Interpretability:** Many machine learning models operate as "black boxes," making it difficult to understand how they arrive at specific decisions. This opacity can hinder trust and accountability [47].
- **Overfitting and Underfitting:** Overfitting occurs when a model learns the training data too well, including details that don’t matter (like noise or outliers). Underfitting happens when a model is too simple and fails to capture the underlying patterns in the data. Both issues can significantly degrade model performance [47].

- **High Computational Costs:** Training and deploying machine learning models can be resource-intensive, requiring significant computational power and energy consumption [47].
- **Data Privacy Concerns:** Machine learning often involves processing sensitive personal data, raising concerns about data privacy and the potential misuse of information [47].

2.4 Deep Learning

Deep learning is a specialized field within machine learning that utilizes neural networks with many layers to learn from large amounts of data(see Figure 2.1)[48]. These networks, often referred to as deep neural networks, are designed to automatically learn hierarchical representations of data. The 'deep' in deep learning refers to the presence of multiple hidden layers between the input and output layers, which allows the model to learn progressively more abstract features from the raw input. Deep learning has shown remarkable success in a wide range of applications, such as image recognition, natural language processing, and autonomous systems, due to its ability to model complex patterns and make highly accurate predictions from vast and unstructured datasets [49].

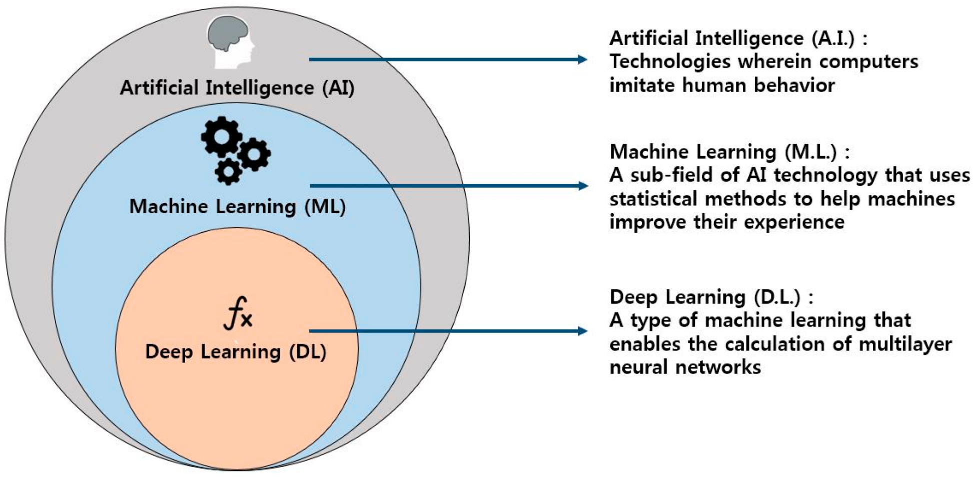

Artificial Intelligence (AI)
Machine Learning (ML)
Deep Learning (DL)
Artificial Intelligence (A.I.) : Technologies wherein computers imitate human behavior
Machine Learning (M.L.) : A sub-field of AI technology that uses statistical methods to help machines improve their experience
Deep Learning (D.L.) : A type of machine learning that enables the calculation of multilayer neural networks

Figure 2.1: Relationship between AI, ML, DL

2.4.1 Neural Networks

Artificial Neural Networks (ANNs), often simply called neural networks (NNs), are a cornerstone of modern machine learning and artificial intelligence. Inspired by the structure and function of the human brain, NNs are computational models composed of interconnected processing units called neurons, organized in layers [50](see Figure 2.2). These networks are particularly adept at identifying complex patterns and relationships within data, making them suitable for a wide array of tasks such as image recognition, natural language processing, and predictive modeling [51]. The fundamental idea is that NNs learn from data by adjusting the strengths of connections (weights) between neurons, much like biological brains learn through synaptic plasticity.

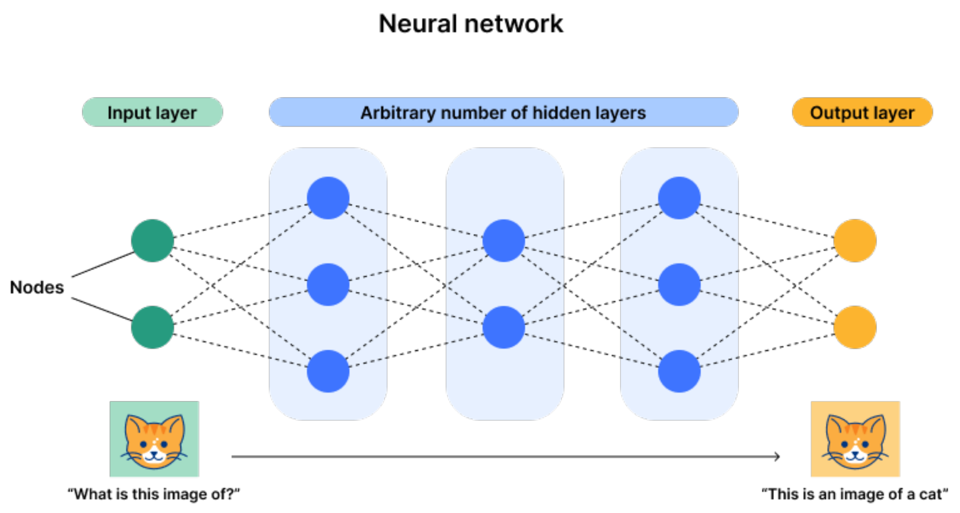


Figure 2.2: Illustration of a neural network

2.4.2 Components of Neural Networks

Neural networks are constructed from several key building blocks. These components define the architecture and the fundamental operations within the network.

Neurons (Nodes or Units)

A neuron (or node) is the fundamental processing unit of a neural network. It receives one or more input signals, processes them, and produces an output signal. Each input signal is associated with a weight, which signifies its importance. The neuron computes a weighted sum of its inputs and adds a bias term. This sum is then passed through an activation function to produce the neuron's output [50].

Mathematically, if a neuron has inputs $x_1, x_2, \ldots, x_n$, corresponding weights:

$$w_1, w_2, \ldots, w_n,$$

and a bias b, the weighted sum z is:

$$z = (w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) + b = \sum_{i=1}^{n} w_i x_i + b \tag{2.1}$$

The output a of the neuron is then:

$$a = f(z) \tag{2.2}$$

where f is the activation function.

Weights

Weights are crucial parameters within the neural network, associated with each connection between neurons. They represent the strength of the connection between units. During the training process, the network learns these weights. A higher absolute weight means that the input from that connection will have a more significant influence (either excitatory or inhibitory) on the neuron's output [50].

Bias

A bias term is an additional parameter associated with each neuron (typically in hidden and output layers). It allows the activation function to be shifted to the left or right, which is critical for the model to fit the data properly. Without a bias, the neuron's activation would only pass through the origin if the weighted sum of inputs is zero, limiting the flexibility of the model [50].

Layers

Neurons are organized into layers. A typical feedforward neural network consists of three types of layers [51]:

1. **Input Layer:** This layer receives the raw input data (features) for the network. The number of neurons in this layer corresponds to the number of features in the input data. It doesn't perform any computation; it simply passes the data to the first hidden layer.

2. **Hidden Layer(s):** These layers are located between the input and output layers. They perform most of the computations and are responsible for learning the complex patterns in the data by transforming the inputs into a space where they become more separable or predictable. A neural network can have zero or more hidden layers. Networks with multiple hidden layers are referred to as *deep* neural networks [52]. The "hidden" aspect refers to the fact that their outputs are not directly observed as inputs or outputs of the overall system.

3. **Output Layer:** This layer produces the final output of the network. The number of neurons and the activation function used in the output layer depend on the type of task (e.g., one neuron with a sigmoid function for binary classification, multiple neurons with a softmax function for multi-class classification, or one neuron with a linear activation for regression).

Data flows from the input layer, through the hidden layer(s), to the output layer in a *feedforward* manner, meaning connections generally flow in one direction without forming cycles [50].

Activation Functions

Activation functions introduce non-linearity into the neural network, which is crucial for learning complex relationships that linear models cannot capture [51]. Without non-linear activation functions, a multi-layer neural network would behave like a single-layer linear model, regardless of its depth.

Common activation functions include:

- **Sigmoid (Logistic) Function**
  - **Formula:**
$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$
  - **Output range:** $(0, 1)$
  - **Properties:** Smooth, differentiable, and commonly used in the output layer for binary classification problems to represent probabilities [50].
  - **Limitations:** Suffers from the *vanishing gradient* problem, where gradients become very small for large positive or negative inputs, slowing down learning, especially in deep networks [51].

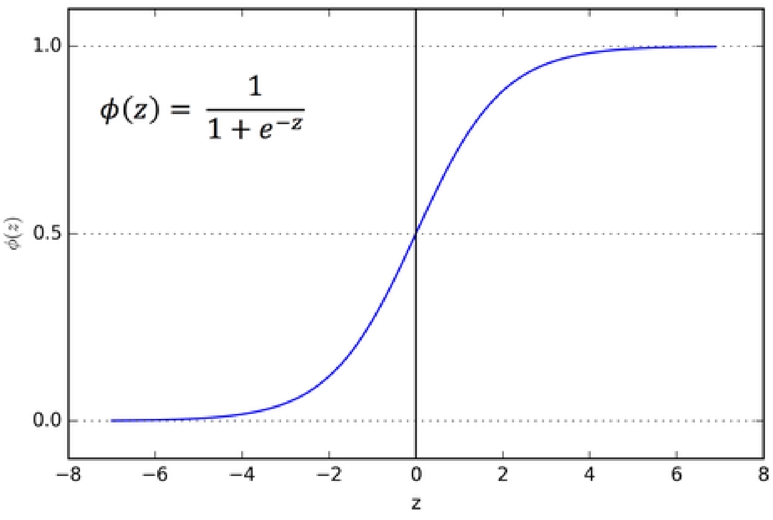


Figure 2.3: Sigmoid activation function

- **Rectified Linear Unit (ReLU)**
  - **Formula:**
$$\text{ReLU}(z) = \max(0, z) \tag{2.4}$$
  - **Output range:** $[0, \infty)$
  - **Properties:** Computationally efficient and helps mitigate the vanishing gradient problem for positive inputs, leading to faster training in many cases [53]. It has become the default activation function for hidden layers in many deep learning applications [52].
  - **Limitations:** Can suffer from the *dying ReLU* problem, where neurons can become inactive and output zero for all inputs if their weights are updated such that z is always negative during training. Variations like Leaky ReLU or Parametric ReLU (PReLU) address this [54].

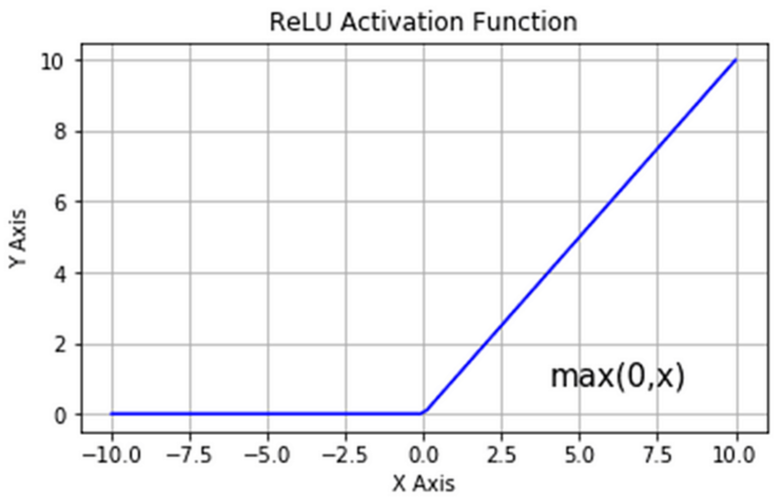


Figure 2.4: Rectified Linear Unit (ReLU) activation function

- **Hyperbolic Tangent (Tanh)**
  - **Formula:**
$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.5}$$
  - **Output range:** $(-1, 1)$
  - **Properties:** Similar to sigmoid but zero-centered, which can sometimes help in learning.
  - **Limitations:** Also suffers from the vanishing gradient problem.

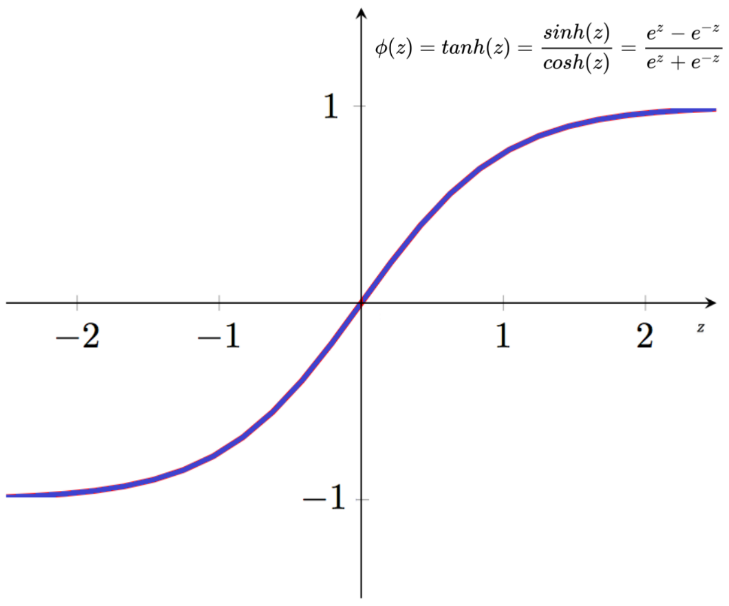


Figure 2.5: Hyperbolic Tangent (Tanh) activation function

Connections (Edges)

Connections are the pathways between neurons that transmit signals. Each connection has an associated weight. In a feedforward network, these connections typically link neurons from one layer to the neurons in the subsequent layer. The pattern of connections defines the architecture of the network [50].

2.4.3 Learning and Training Neural Networks

The process by which a neural network "learns" is through training on a dataset. This involves iteratively adjusting the network's weights and biases to minimize the difference between its predictions and the actual target values.

Forward Propagation

Forward propagation is the process of passing an input data sample through the network, from the input layer to the output layer, to generate a prediction [50]. For each layer:

1. The weighted sum of inputs (plus bias) from the previous layer is calculated for each neuron.
2. This sum is then passed through the neuron's activation function.

The outputs of the current layer become the inputs for the next layer. This continues until the output layer produces the network's prediction for the given input.

Loss Functions

A loss function (or cost function) quantifies how well the network's predictions match the actual target values from the training data. The goal of training is to minimize this loss [51]. The choice of loss function depends on the type of problem:

- **Mean Squared Error (MSE):**
  - **Formula:**
  $$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.6}$$
  - **Where:**
    - N is the number of samples,
    - y_i is the true target value,
    - $\hat{y}_i$ is the predicted value.
  - **Usage:** Commonly used for regression tasks where the goal is to predict continuous values [50].
- **Cross-Entropy Loss (or Log Loss):**
  - **Binary Cross-Entropy Formula:**
  $$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{2.7}$$

  - **Categorical Cross-Entropy:** A generalization for multi-class classification, often used in conjunction with the Softmax function in the output layer.
  - **Usage:** Commonly used for classification tasks. It measures the performance of a classification model whose output is a probability value between 0 and 1 [52]. It penalizes confident but incorrect predictions more heavily.

Backpropagation

Backpropagation, short for "backward propagation of errors," is the core algorithm used to train neural networks [55]. After calculating the loss using forward propagation, backpropagation efficiently computes the gradient of the loss function with respect to each weight and bias in the network. The process involves:

1. Calculating the error at the output layer.
2. Propagating this error backward, layer by layer, from the output layer to the input layer.

At each layer, it calculates how much each neuron's weights and bias contributed to the overall error, using the chain rule of calculus [50]. These gradients indicate the direction and magnitude of change needed for each weight and bias to reduce the loss.

Optimization Algorithms and Learning Rate

Once the gradients are computed via backpropagation, an optimization algorithm is used to update the weights and biases of the network in a way that minimizes the loss function.

- **Learning Rate (η):** A crucial hyperparameter that controls the step size during the weight update process in optimization algorithms such as gradient descent. The update rule for a weight w is:

$$w = w - \eta \cdot \frac{\partial L}{\partial w} \tag{2.8}$$

  where:

  - w is the weight parameter being updated,
  - η is the learning rate,
  - L is the loss function,
  - $\frac{\partial L}{\partial w}$ is the partial derivative of the loss function with respect to w, indicating the direction and magnitude of change needed.

  A learning rate that is too small can lead to very slow convergence, while a learning rate that is too large can cause the optimization process to overshoot the minimum or diverge entirely.

- **Gradient Descent (GD):**

  - The basic idea is to take steps in the direction opposite to the gradient of the loss function [50].
  - Batch Gradient Descent calculates the gradient using the entire training dataset, which can be computationally expensive for large datasets.

- **Stochastic Gradient Descent (SGD):**
  - Updates the parameters using the gradient calculated from a single training sample (stochastic) or a small batch of samples (mini-batch SGD) at a time [56].
  - This makes training faster and can help escape shallow local minima due to the noisy updates [51]. Mini-batch SGD is the most common approach.

- **Adam (Adaptive Moment Estimation):**
  - An adaptive learning rate optimization algorithm that combines ideas from RMSprop and Momentum [57].
  - It computes adaptive learning rates for each parameter by keeping track of exponentially decaying averages of past gradients (first moment) and past squared gradients (second moment).
  - Adam is often effective and a popular default choice for many deep learning applications [57].

Other optimizers include Adagrad, RMSprop, and AdaDelta, each with different strategies for adapting learning rates.

2.5 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural networks specifically designed to process sequential data, such as text, speech, or time series, where the order of information is crucial. Unlike feedforward networks that process inputs independently, RNNs have connections that form directed cycles, allowing them to maintain an internal state or "memory" of past information to influence current and future predictions [52].

2.5.1 Architecture and Mechanism of RNNs

The core idea behind Recurrent Neural Networks (RNNs) is the use of a recurrent connection in their hidden layers. At each time step t, a neuron in an RNN receives not only the current input x_t but also the hidden state h_{t-1} from the previous time step [50]. This hidden state acts as a summary of the information processed from previous steps in the sequence.

The hidden state at time step t, denoted h_t, is typically computed as:

$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{2.9}$$

where:

- x_t is the input at time step t,

- h_{t-1} is the hidden state from the previous time step,
- W_{hh} is the weight matrix for the recurrent (hidden-to-hidden) connection,
- W_{xh} is the weight matrix for the input-to-hidden connection,
- b_h is the bias for the hidden layer,
- f is an activation function (e.g., *Tanh* or *ReLU*).

The output at time step t, denoted y_t, is computed from the current hidden state:

$$y_t = g(W_{hy}h_t + b_y) \tag{2.10}$$

where:

- W_{hy} is the weight matrix for the hidden-to-output connection,
- b_y is the bias for the output layer,
- g is an activation function for the output layer (e.g., *Softmax* for classification).

The same set of weights (W_{hh}, W_{xh}, W_{hy}) and biases (b_h, b_y) are used across all time steps. This parameter sharing enables RNNs to generalize to sequences of varying lengths and to recognize patterns that appear at different positions within the sequence [50].

For purposes of training and explanation, RNNs are often visualized as being "unrolled" or "unfolded" through time, effectively transforming their recurrent structure into a deep feedforward network where each layer corresponds to a specific time step (see Figure 2.6) [50].

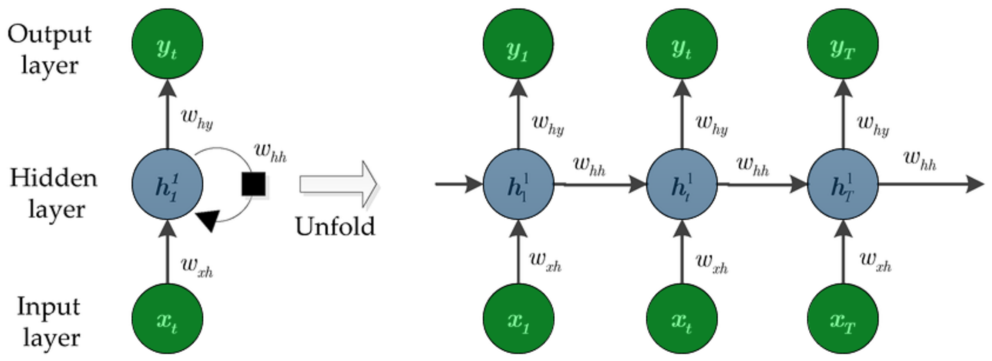


Figure 2.6: Architecture of a Recurrent Neural Network (RNN)

2.5.2 Limitations and Challenges

Training Recurrent Neural Networks (RNNs) involves a modified version of backpropagation known as Backpropagation Through Time (BPTT) [58]. BPTT unrolls the network and sums gradients across all time steps. However, this process introduces several significant challenges, particularly with long sequences:

1. **Vanishing Gradients:**

- **Problem:** During BPTT, gradients are propagated backward through many time steps. If the recurrent weight matrix W_{hh} (or more accurately, the Jacobian of the hidden state transition) has leading eigenvalues less than 1, the gradients can shrink exponentially as they are propagated back through time [59].
- **Consequence:** This makes it difficult for the network to learn long-range dependencies, as the influence of earlier inputs on later outputs (and thus on the loss) becomes negligible. The network effectively forgets information from the distant past [58].
- **Impact:** The model struggles to capture relationships between elements that are far apart in a sequence.

2. **Exploding Gradients:**
    - **Problem:** Conversely, if the leading eigenvalues of the Jacobian of the hidden state transition are greater than 1, the gradients can grow exponentially as they are propagated backward [59].
    - **Consequence:** This leads to very large updates to the weights, causing unstable training. The loss can become NaN (Not a Number), and the model may diverge.
    - **Mitigation:** A common technique to counteract exploding gradients is gradient clipping, where gradients are scaled down if their norm exceeds a predefined threshold [60].
3. **Difficulty Learning Long-Term Dependencies:**
    - This is a direct consequence, primarily of the vanishing gradient problem but also related to the way information is compressed into the hidden state over many steps [58]. Even if gradients don't completely vanish, they might not be informative enough to capture subtle, long-range contextual information.

These challenges have motivated the development of more sophisticated RNN architectures like Long Short-Term Memory (LSTM) [61] and Gated Recurrent Units (GRU) [62], which use gating mechanisms to better control the flow of information and mitigate the vanishing/exploding gradient problems, enabling them to learn much longer dependencies.

2.6 Recurrent Neural Network Variants

While standard Recurrent Neural Networks (RNNs) are designed to process sequential data, they often suffer from the vanishing or exploding gradient problem, especially when dealing with long-range dependencies [59]. This limitation makes it difficult for vanilla RNNs to learn and retain information over extended time steps. To address this, more sophisticated architectures like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) were developed. These networks incorporate gating mechanisms that allow them to selectively remember or forget information, thereby mitigating the gradient issues and enabling the learning of long-term dependencies more effectively [59].

2.6.1 Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) network, introduced by Hochreiter and Schmidhuber in 1997 [61], is a specialized type of RNN explicitly designed to learn long-term dependencies. The core innovation of LSTMs lies in their memory cell (or cell state) and a series of gates that regulate the flow of information into and out of this cell.

The architecture of an LSTM cell (Figure 2.7) typically consists of three main gates [59]:

1. **Forget Gate** (f_t): This gate decides what information should be discarded from the cell state. It looks at the previous hidden state h_{t-1} and the current input x_t, and outputs a number between 0 and 1 for each element in the cell state C_{t-1}. A value of 1 represents "completely keep this" while 0 means "completely get rid of this".

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.11}$$

2. **Input Gate** (i_t): This gate determines which new information will be stored in the cell state. It has two parts:

   - A sigmoid layer (the input gate) decides which values to update:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2.12}$$

   - A tanh layer creates a vector of new candidate values $\tilde{C}_t$:

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{2.13}$$

   These two components are then combined to update the cell state.

3. **Output Gate** (o_t): This gate determines the next hidden state h_t. The output is based on the cell state but is a filtered version.

   - A sigmoid layer decides which parts of the cell state to output:

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{2.14}$$

   - The cell state is passed through a tanh function and multiplied by the output gate:

$$h_t = o_t \cdot \tanh(C_t) \tag{2.15}$$

The cell state C_t is updated by first forgetting some information (multiplying the old cell state C_{t-1} by f_t) and then adding the new candidate values scaled by the input gate ($i_t \cdot \tilde{C}_t$):

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{2.16}$$

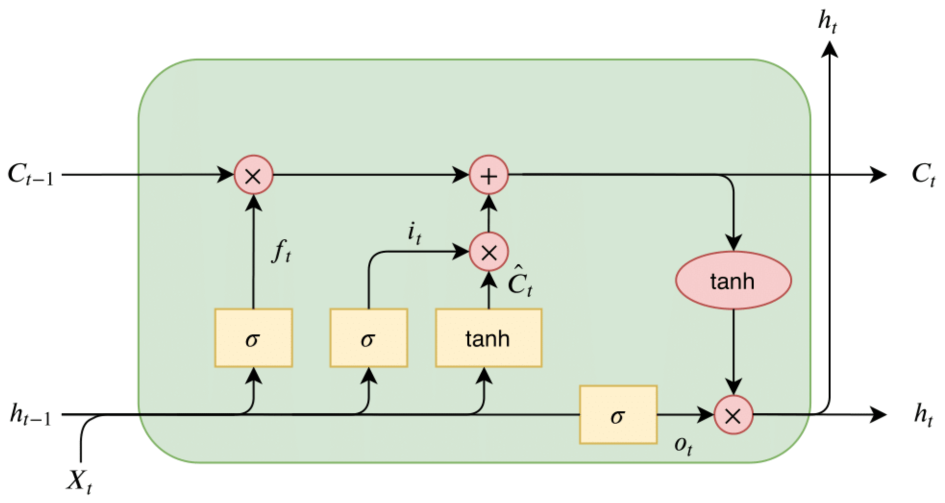


Figure 2.7: Long Short-Term Memory (LSTM) Architecture

2.6.2 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU), introduced by Cho et al. (2014) [62], is a more recent advancement that simplifies the LSTM architecture while often providing comparable performance [59]. GRUs combine the forget and input gates into a single *update gate* and also merge the cell state and hidden state.

The architecture of a GRU cell (Figure 2.8) features two primary gates [59]:

1. **Update Gate** (z_t): This gate determines how much of the past information from previous time steps should be carried forward. It essentially combines the roles of the forget and input gates in an LSTM, deciding what information to discard and what to retain.

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t] + b_z\right) \tag{2.17}$$

2. **Reset Gate** (r_t): This gate controls how much of the past information to forget. It is applied to the previous hidden state before computing the new candidate hidden state, allowing the model to selectively ignore past context.

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t] + b_r\right) \tag{2.18}$$

Next, the **candidate hidden state** $\tilde{h}_t$ is computed using the reset gate:

$$\tilde{h}_t = \tanh\left(W_h \cdot [r_t * h_{t-1}, x_t] + b_h\right) \tag{2.19}$$

Finally, the current hidden state h_t is calculated as a linear interpolation between the previous hidden state h_{t-1} and the candidate hidden state $\tilde{h}_t$, controlled by the update gate:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{2.20}$$

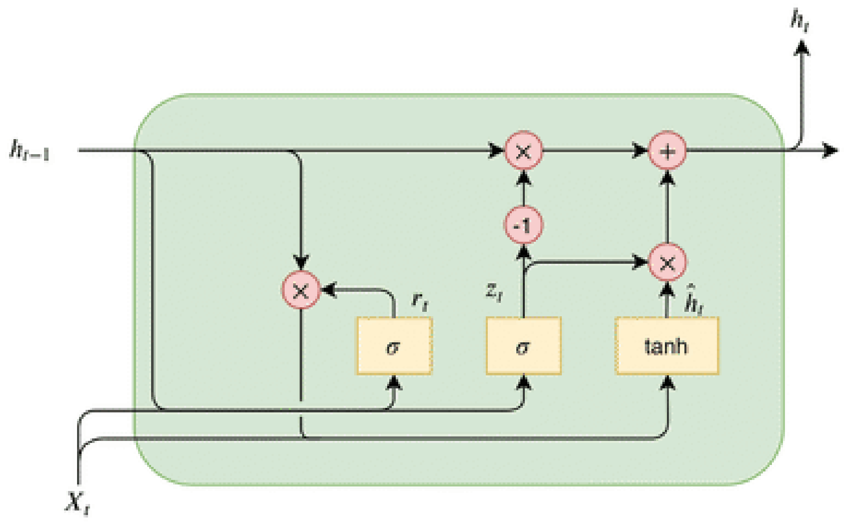


Figure 2.8: Gated Recurrent Unit (GRU) Architecture

2.7 Conclusion

In this chapter, we introduced the fundamental concepts behind Recurrent Neural Networks (RNNs) and their role in processing sequential data. Starting with a brief overview of artificial intelligence, machine learning, and deep learning, which gave the background information needed to examine neural network architectures. The focus then shifted to RNNs, highlighting their ability to model temporal dependencies and their relevance to tasks where the order of inputs matters such as in session-based recommendation systems.

Understanding the mechanisms behind RNNs and their variants particularly LSTM, provides a solid theoretical foundation for the personalized recommendation model proposed in this thesis. This information will be used directly in the following chapter and is essential for tackling the difficulties of modeling dynamic user behavior.

Chapter 3

A Session-based Recommender System Using LSTM for E-commerce

3.1 Introduction

In the rapidly evolving landscape of e-commerce, delivering personalized recommendations has become essential for enhancing user experience and increasing sales. Traditional recommender systems frequently depend on long-term user profiles and historical data, which may be unavailable for new or anonymous users. To overcome this limitation, session-based recommender systems have been introduced as an effective solution by concentrating exclusively on the user's current interaction session.

This chapter explores the design and implementation of a session-based recommender system using Long Short-Term Memory (LSTM) networks, a type of recurrent neural network which are ideal for modeling sequential data. Unlike conventional approaches, LSTM-based models are able to efficiently capture the short-term dependencies and patterns in user behavior within a session, enabling more accurate and timely product recommendations.

3.2 Deep Learning Approaches

Deep learning has significantly advanced the field of recommender systems by addressing many limitations of traditional methods. Its capacity to model complex data and learn meaningful representations has made it a popular choice in modern recommendation tasks. The key strengths of deep learning-based recommender systems were outlined as follows in [63]:

- **Nonlinear Transformation:**
  Deep neural networks can model complex, nonlinear user-item interactions through nonlinear activation functions (e.g., ReLU, sigmoid, tanh). This allows them to capture intricate patterns that linear models like matrix factorization or factorization machines cannot effectively represent.

- **Representation Learning:**
  Deep learning models automatically learn useful features from raw data, reducing the need for manual feature engineering. They can also incorporate diverse content types such as text, images, audio, and video, enhancing the quality of user and item representations.

- **Sequence Modeling:**
  Techniques such as RNNs and CNNs are well-suited for modeling sequential data, enabling the capture of temporal dynamics in user behavior and item evolution. This capability is critical for tasks like next-item prediction and session-based recommendation.

- **Flexibility:**
  The availability of powerful deep learning frameworks (e.g., TensorFlow, Keras, PyTorch) allows for flexible model design and easy experimentation, facilitating the application of deep learning across various recommendation scenarios.

3.3 A Session-based Recommender System Using LSTM

Long Short-Term Memory (LSTM) network offers a robust approach for session-based recommender systems, enabling the generation of item suggestions based on a user's current session of interactions. LSTM is particularly effective at capturing the sequential dependencies and temporal patterns within session data. By learning from the order of item interactions, the LSTM-based model can predict the next likely item in a session, even in cases where long-term user histories are unavailable. This makes it highly suitable for e-commerce environments, where many users interact anonymously or for short durations.

3.3.1 Architecture

LSTM networks offer a principled approach to modeling session-based recommendation tasks, particularly suited to sequential interaction data where long-term user histories may be sparse or unavailable. The proposed architecture treats each session as an ordered sequence of item interactions and aims to predict the next item based on historical context.

Let a user session be represented as:

$$\mathbf{s} = [i_1, i_2, ..., i_t], \quad i_k \in \{1, ..., N\} \tag{3.1}$$

- $\mathbf{s}$ denotes a user session as an ordered sequence of items.
- i_k is the item index at timestep k.
- N is the total number of unique items in the dataset.
- t is the length of the session (i.e., the number of items interacted with).

1. **Input Layer**
   The input to the model is the sequence $\mathbf{s}$ of item indices. Since sessions may vary in length, padding is applied using a special token (zero) to standardize input length to a fixed maximum T:
   $$\mathbf{s} \in \mathbb{Z}^T \tag{3.2}$$
   - T is the maximum session length.
   - Padding facilitates batch processing.
   - A masking mechanism is employed to ignore padding tokens during training and evaluation.

2. **Embedding Layer**
   Each item index i_k is mapped to a dense vector via an embedding lookup:
   $$\mathbf{x}_k = \mathbf{E}[i_k] \tag{3.3}$$
   - $\mathbf{E} \in \mathbb{R}^{N \times d}$ is the embedding matrix.
   - d is the embedding dimension (a hyperparameter).

- The sequence of embeddings is: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]$

This layer transforms sparse categorical inputs into dense continuous vectors that capture latent item features.

3. **LSTM Layer**
The sequence of embedding vectors $\mathbf{X}$ is passed through an LSTM layer to capture temporal dependencies:
$$\mathbf{h}_k = \text{LSTM}(\mathbf{x}_k, \mathbf{h}_{k-1}) \tag{3.4}$$
   - $\mathbf{h}_k \in \mathbb{R}^h$ is the hidden state at timestep k.
   - h is the number of LSTM hidden units (a hyperparameter).
   - The final hidden state $\mathbf{h}_T$ summarizes the session context.

4. **Output Layer**
The final hidden state $\mathbf{h}_T$ is passed through a fully connected layer followed by a softmax activation to produce the probability distribution over all possible next items:
$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}\mathbf{h}_T + \mathbf{b}) \tag{3.5}$$
   - $\mathbf{W} \in \mathbb{R}^{N \times h}$ is the weight matrix.
   - $\mathbf{b} \in \mathbb{R}^N$ is the bias vector.
   - $\hat{\mathbf{y}} \in [0, 1]^N$ is the predicted probability vector for the next item.

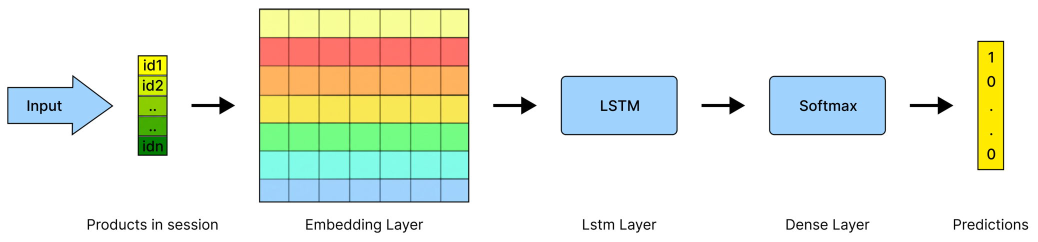


Figure 3.1: Architecture of the proposed LSTM model

3.3.2 Training

The training configuration was designed to effectively optimize model parameters while enhancing generalization, ensuring convergence, and reducing the risk of overfitting.

Loss Function

The model is trained using the *sparse categorical cross-entropy* loss, which is suitable for multi-class classification tasks where the target labels are integer indices. This loss measures the discrepancy between the predicted probability distribution and the true class label. It is defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{y}_{i,t}) \tag{3.6}$$

where:

- N is the number of training examples.
- t is the correct class index for the i-th sample.
- $\hat{y}_{i,t}$ is the predicted probability assigned to the correct class.

This formulation ensures that the model is penalized when it assigns low probability to the correct next item in the sequence.

Optimization Algorithm

To minimize the loss, the model employs the *Adam optimizer*, an adaptive learning rate method that combines the benefits of AdaGrad and RMSProp. Adam adjusts the learning rate individually for each parameter using estimates of the first and second moments of the gradients.

The update rule is given by:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.7}$$

where:

- η_t is the learning rate at time step t,
- $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected estimates of the first and second moments of the gradient,
- ϵ is a small constant to prevent division by zero.

To ensure stable convergence, the learning rate follows an exponential decay schedule:

$$\eta_t = \eta_0 \cdot \gamma^{\frac{t}{s}} \tag{3.8}$$

with:

- Initial learning rate $\eta_0 = 0.001$,
- Decay rate $\gamma = 0.9$,
- Decay steps $s = 1000$.

Regularization

To enhance generalization and reduce the risk of overfitting, several regularization strategies are employed:

- **Dropout:** A dropout rate is applied after the LSTM layer to randomly deactivate a fraction of neurons during training, thereby preventing co-adaptation.
- **Embedding Diversity Regularizer:** This regularizer promotes diversity among the learned embeddings by penalizing their pairwise cosine similarity. Given the embedding matrix $X \in \mathbb{R}^{n\times d}$, where n is the number of items and d is the embedding dimension, the regularization loss is computed as follows:
  - Each embedding vector $\mathbf{v}_i$ is normalized to unit length:
    $$\hat{\mathbf{v}}_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\| + \epsilon} \tag{3.9}$$
    where ϵ is a small constant to prevent division by zero.
  - The cosine similarity matrix S is computed as:
    $$S = \hat{X} \cdot \hat{X}^T \tag{3.10}$$
    where each element S_{ij} represents the cosine similarity between embeddings $\hat{\mathbf{v}}_i$ and $\hat{\mathbf{v}}_j$.
  - The diversity loss is defined as the sum of squared cosine similarities across all pairs:
    $$L_{diversity} = \sum_{i=1}^{n}\sum_{j=1}^{n} S_{ij}^2 \tag{3.11}$$
  - Finally, the regularization term added to the overall loss is weighted by a hyper-parameter λ:
    $$L_{reg} = \lambda \cdot L_{diversity} \tag{3.12}$$

  This approach encourages the model to learn more diverse and informative embeddings, improving representational richness.
- **Early Stopping:** Training is halted if the model's validation loss does not improve after a certain number of consecutive epochs (patience). This prevents overfitting by stopping the training process before the model starts to memorize noise in the training data.

  Formally, if the validation loss L_{val} does not improve for p epochs, training stops early.

Hyperparameters

The table below summarizes the key hyperparameters used during training:

Hyperparameter	**Value**
Embedding dimension	512
LSTM units	300
Dropout rate	0.7
Batch size	512
Maximum epochs	16
Early stopping patience	5 epochs

Table 3.1: Summary of training hyperparameters

3.4 Development Tools and Environment

The implementation of this work relied on a range of well-established libraries and cloud-based platforms that enabled scalable experimentation and training. These tools provided essential functionality for data preprocessing, model design, training, and visualization, while the computational environments provided the necessary resources for high-performance development and testing.

Python

Python[1] is a high-level programming language widely adopted in the scientific and machine learning communities due to its readability, simplicity, and extensive ecosystem of libraries. It served as the main language for developing and orchestrating all components of the project.

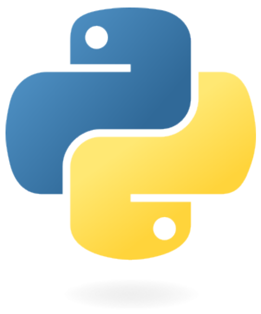


Figure 3.2: Python logo

TensorFlow & Keras

TensorFlow[2] is an open-source deep learning framework developed by Google. It supports scalable model training and deployment across various hardware platforms, including CPUs, GPUs, and TPUs, making it highly flexible for different computing environments. Keras[3], integrated within TensorFlow, provides a user-friendly, high-level API that simplifies the construction and training of deep learning models, making it easier to experiment and develop solutions efficiently.

[1] `https://www.python.org`
[2] `https://www.tensorflow.org`
[3] `https://keras.io`

Figure 3.3: TensorFlow logo

Figure 3.4: Keras logo

NumPy

NumPy[4] is a fundamental Python library for numerical computing that provides powerful n-dimensional array objects optimized for efficient storage and fast computation. It offers a wide range of mathematical functions for operations like linear algebra, statistics, and Fourier transforms, enabling high-performance processing of large datasets. NumPy's ability to perform vectorized operations and broadcasting makes it essential for handling high-dimensional data in machine learning and scientific applications. It also integrates smoothly with other libraries and frameworks, making it a cornerstone of the Python data ecosystem.

Figure 3.5: NumPy logo

Pandas

Pandas[5] is an open-source Python library that provides high-performance, easy-to-use data structures and data analysis tools. It introduces two primary data structures: Series (1-dimensional) and DataFrame (2-dimensional), which are designed to handle a wide range of data types and formats. Pandas excels at managing structured data, offering functionalities such as intelligent label-based indexing, handling of missing data, and support for time series operations. It facilitates tasks like data cleaning, transformation, aggregation, and visualization, making it an essential tool in data science, analytics, and machine learning workflows.

[4] `https://numpy.org`

[5] `https://pandas.pydata.org`

Figure 3.6: Pandas logo

Matplotlib

Matplotlib[6] is a plotting library for Python that supports the creation of static, animated, and interactive visualizations. It allows users to generate a wide range of plots such as line graphs, histograms, scatter plots, and bar charts, with extensive options for customizing the appearance of figures. It works well with libraries like NumPy and Pandas, making it suitable for visualizing data throughout the analysis pipeline. In machine learning, it is commonly used to display training metrics like loss and accuracy curves to monitor and evaluate model performance over time.

Figure 3.7: Matplotlib logo

Kaggle

Kaggle[7] is an online platform that serves as a hub for data science and machine learning enthusiasts. It offers a collaborative environment where users can participate in competitions, access a vast repository of public datasets, and utilize cloud-based notebooks for developing and sharing code. Kaggle's infrastructure supports various computational resources, including GPUs and TPUs, facilitating efficient model training and experimentation. The platform also provides educational resources to help users enhance their data science skills.

Figure 3.8: Kaggle logo

Google Cloud TPUs

Google Cloud TPUs[8] are custom-designed application-specific integrated circuits (ASICs) developed by Google to accelerate machine learning workloads. Optimized for large-scale matrix computations, TPUs are particularly effective for training and inference of complex

[6] `https://matplotlib.org`
[7] `https://www.kaggle.com`
[8] `https://cloud.google.com/tp`

models, including large language models and recommendation systems. Cloud TPUs support popular machine learning frameworks such as TensorFlow, PyTorch, and JAX, providing scalable and efficient resources for AI development. By leveraging Kaggle's free TPU access, this project was able to significantly reduce training time while handling large datasets efficiently.

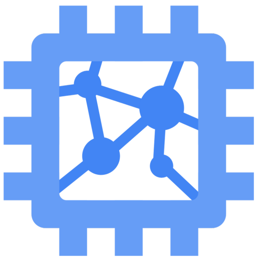


Figure 3.9: Google Cloud TPUs logo

3.5 Experiments and Analysis

This section describes the experimental setup used to evaluate the performance of the proposed LSTM-based model. It includes an overview of the datasets and preprocessing steps, a description of the evaluation metrics, and a comparative analysis against baseline models to highlight the improvements achieved.

3.5.1 Datasets

Yoochoose

Yoochoose[9] is a public dataset released as part of the RecSys Challenge 2015. It contains anonymized clickstream data collected from an e-commerce website over a six-month period. Each record consists of a session ID, timestamp, and item ID, enabling the construction of sequential user interaction data. For our experiments, we used both the 1/64 and 1/4 subsets of the dataset. These subsets are widely adopted in related work due to their balance between dataset size and session diversity, allowing for effective training and evaluation under different computational constraints.

Diginetica

Diginetica[10] is a public dataset released as part of the CIKM Cup 2016 Challenge and contains anonymized search and browsing logs, product metadata, transaction histories, and a large set of product images. It reflects real-world e-commerce behavior, including both "query-full" and "query-less" sessions. The goal is to predict product relevance based on individual user shopping, search, and browsing preferences. For session-based recommendation tasks, the dataset is typically filtered to focus on click and transaction sequences, offering a rich and challenging benchmark due to its sparsity and varied user behavior patterns.

[9] `https://www.kaggle.com/datasets/chadgostopp/recsys-challenge-2015>`
[10] `https://competitions.codalab.org/competitions/11161`

Data Preprocessing

To ensure high-quality input for training and evaluation, both datasets underwent a series of preprocessing steps, following the methodology outlined in [64]:

- Filtered out sessions with fewer than two interactions to remove very short and uninformative sequences.
- Excluded items that appeared fewer than five times across the dataset to reduce noise and improve model generalization.
- Split the remaining sessions chronologically into training and test sets, using the last day of interactions for the Yoochoose dataset as the test set, and the last week for the Diginetica dataset.
- Removed items that appeared only in the test set but not in the training set to maintain consistency.
- Mapped all remaining item IDs to continuous integer indices to enable embedding and ensure compatibility with the model.

These preprocessing steps resulted in a clean, compact, and representative dataset for evaluating session-based recommendation models. A summary of the dataset's properties after preprocessing is provided in Table 3.2.

Dataset	# of Clicks	# of Sequences	# of Items	Avg. Length
Yoochoose 1/64 **(Train)**	486,614	369,859	17,370	6.39
Yoochoose 1/4 **(Train)**	7,841,282	5,917,745	30,445	6.02
Yoochoose **(Test)**	71,222	55,898	6,751	6.72
Diginetica **(Train)**	906,140	719,470	43,097	5.39
Diginetica **(Test)**	76,821	60,858	21,131	5.30

Table 3.2: Statistics of Datasets Used in the Experiments

3.5.2 Evaluation Metrics

To assess the performance of the session-based recommendation model, the following evaluation metrics were used:

Accuracy

Accuracy measures the proportion of correct predictions made by the model out of all predictions. In the context of recommendation, it usually refers to how often the top recommended item matches the actual next item in the session.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{3.13}$$

Mean Reciprocal Rank at 20 (MRR@20)

MRR@20 evaluates the average of the reciprocal ranks of the true next item within the top 20 recommendations. It gives higher scores when the correct item appears closer to the top of the recommendation list. For each query/session, if the true item is ranked at position rank_i, then:

$$\text{MRR@20} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i} \tag{3.14}$$

where $\text{rank}_i \leq 20$, otherwise the reciprocal rank is zero.

Hit Rate at 20 (HR@20)

Hit Rate@20 measures whether the true next item appears anywhere in the top 20 recommended items. It is a binary measure per session — 1 if the true item is in the top 20 recommendations, 0 otherwise.

$$\text{HR@20} = \frac{1}{N}\sum_{i=1}^{N}\text{hit}_i \tag{3.15}$$

where:

$$\text{hit}_i = \begin{cases} 1 & \text{if true item is in top 20 recommendations} \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

These metrics together provide a comprehensive evaluation by measuring not only the correctness of predictions (Accuracy) but also their rank quality (MRR@20) and recall within the top recommendations (HR@20).

3.5.3 Baselines

To evaluate the performance of the proposed model, it was compared with several baseline methods commonly used in session-based recommendation systems.

- **POP:** Recommends the top-k items with the highest overall interaction frequency. It provides the same list of popular items to all users, regardless of their individual preferences or session context.
- **S-POP:** A variation of the POP method that considers the current session. It prioritizes items that appear within the ongoing session, placing them at the top of the recommendation list based on their frequency within that session, followed by globally popular items.
- **Item-KNN:** This approach recommends items that are similar to those the user has interacted with during the current session. Similarity between items is typically computed using cosine similarity, allowing the model to suggest items that frequently co-occur or share similar interaction patterns [65].

- **BPR-MF:** A matrix factorization method that learns latent representations for users and items by optimizing a pairwise ranking objective. In session-based recommendation, a session representation is built by averaging the latent vectors of the items in that session, which is then used to rank unseen items [66].
- **FPMC:** A sequential recommendation model that integrates matrix factorization with Markov chains to capture transition dynamics between items. When applied to session-based scenarios, user identity is ignored, and the model focuses solely on learning item-to-item transitions within sessions [67].
- **GRU4REC:** GRU4REC applies Gated Recurrent Units (GRU) to model entire user sessions in recommender systems, enabling recommendations based on short session data instead of long user histories. It introduces a ranking loss function tailored for session-based recommendation tasks [2].
- **NARM:** NARM combines a hybrid encoder with an attention mechanism to capture both the sequential behavior of users and their main purpose within a session. This unified session representation improves recommendation accuracy, especially for longer sessions. The model uses a bi-linear matching scheme to score candidate items [68].
- **STAMP:** STAMP captures users' general interests through long-term memory of session context while emphasizing their current focus using a short-term attention mechanism. It learns a unified embedding space and incorporates the last-clicked item to better predict the next item. This approach is effective in session-based scenarios lacking user profiles [69].
- **STAN:** STAN extends session-based k-nearest neighbors (SKNN) by incorporating the position of items, recency of past sessions, and position of recommendable items in neighbor sessions. Adjustable decay parameters control these effects, allowing flexibility for different applications [70].
- **CSRM:** CSRM uses two parallel memory modules: an Inner Memory Encoder for modeling the current session's user behavior with RNNs and attention, and an Outer Memory Encoder to incorporate collaborative neighborhood session information. A fusion gating mechanism combines both sources to generate the final session representation [71].

3.5.4 Results and Discussion

Results

The performance results of various sequential recommendation methods, evaluated using Hit Rate at 20 (HR@20) and Mean Reciprocal Rank at 20 (MRR@20), are presented in Table 3.3. These experiments were conducted across three distinct datasets: YOOCHOOSE 1/64, YOOCHOOSE 1/4, and Diginetica. As highlighted in the table, the proposed **LSTM** model consistently demonstrates competitive or superior performance across most metrics and datasets, notably achieving the best results on the YOOCHOOSE datasets. A detailed analysis of these results and their implications follows in the subsequent discussion.

Methods	YOOCHOOSE 1/64		YOOCHOOSE 1/4		Diginetica	
	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20
POP	6.71	1.65	1.33	0.30	0.89	0.20
S-POP	30.44	18.35	27.08	17.75	21.06	13.68
Item-KNN	51.60	21.81	52.31	21.70	35.75	11.57
BPR-MF	30.30	12.08	1.57	1.57	5.24	1.98
FBMC	45.62	15.01	-	-	26.53	6.95
GRU4REC	60.64	22.89	59.53	22.60	29.45	8.33
NARM	68.32	28.63	69.73	29.23	49.70	16.17
STAMP	68.74	29.67	70.44	30.00	45.64	14.32
STAN	69.45	28.74	70.07	28.89	50.97	**18.48**
CSRM	69.85	29.71	70.63	29.48	**51.69**	16.92
Our Model	**70.15**	**31.21**	**71.15**	**31.39**	47.55	15.48

Table 3.3: Performance Comparison of Different Methods

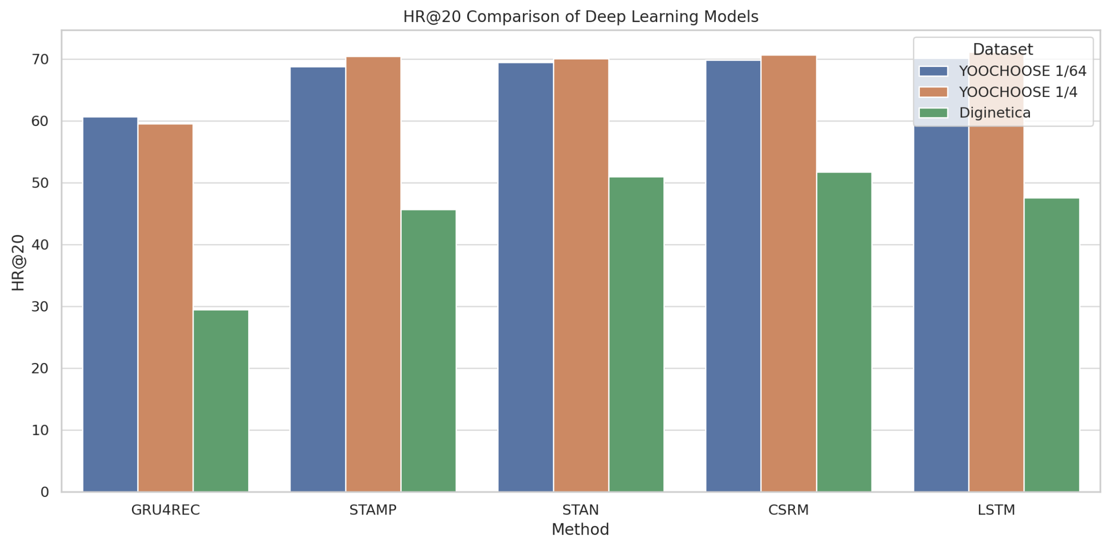


Figure 3.10: Comparison of HR@20 across deep learning methods

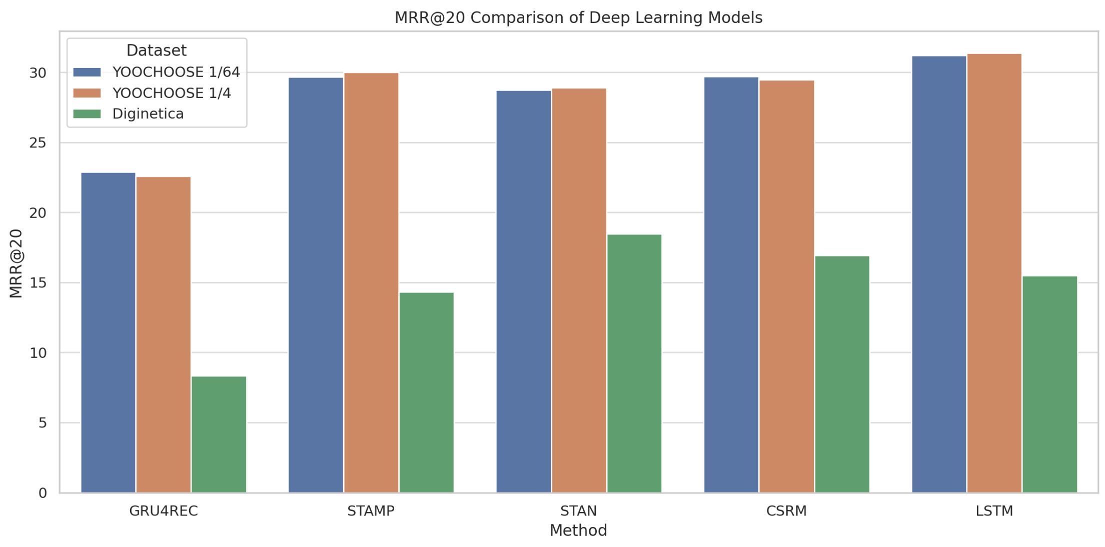


Figure 3.11: Comparison of MRR@20 across deep learning methods

Discussion

This section delves into a detailed analysis of the experimental results presented in Table 3.3, highlighting the strengths and limitations of the proposed **LSTM** model in comparison to established sequential recommendation methods across different datasets.

As expected, deep learning-based sequential models universally outperform traditional baselines such as POP, S-POP, Item-KNN, BPR-MF, and FBMC across all three datasets and metrics. This reinforces the effectiveness of neural networks in capturing complex sequential patterns in user behavior compared to simpler statistical or matrix factorization approaches. Among the traditional methods, Item-KNN generally performs best, indicating the importance of item-to-item similarity in user sessions.

Our proposed **LSTM** model demonstrates remarkable performance on the YOOCHOOSE datasets. On YOOCHOOSE 1/64, it achieves the highest HR@20 of **70.15** and MRR@20 of **31.21**, setting new state-of-the-art results among the compared methods. Similarly, for YOOCHOOSE 1/4, it reaches peak scores of **71.15** for HR@20 and **31.39** for MRR@20. These superior results suggest that **LSTM** effectively captures long-term dependencies and diverse user preferences within these larger and relatively denser datasets. Its ability to incorporate the "diversity" aspect appears particularly beneficial in environments with richer item-interaction histories, enabling more comprehensive and accurate recommendations compared to other deep learning models like NARM, STAMP, STAN, and CSRM.

However, the performance landscape shifts when considering the Diginetica dataset. While **LSTM** still performs significantly better than traditional baselines, it exhibits a noticeable drop in performance (HR@20: **47.55**, MRR@20: **15.48**) relative to its YOOCHOOSE scores. More critically, on Diginetica, other advanced sequential models such as STAN (HR@20: 50.97, MRR@20: 18.48) and CSRM (HR@20: 51.69, MRR@20: 16.92) achieve superior results for HR@20, and STAN for MRR@20. This suggests that **LSTM** might not generalize as effectively to the specific characteristics of this dataset.

This discrepancy can likely be attributed to differences in dataset characteristics. Diginetica is often recognized for being more challenging due to its potentially shorter average session lengths, higher sparsity, or a greater prevalence of novel and cold-start items compared to the YOOCHOOSE datasets. These factors can limit the amount of dense sequential information available for learning intricate patterns. Models heavily relying on capturing long, rich sequences, such as complex LSTM-based architectures, might struggle more in such sparse environments where the "context" for prediction is more limited or fragmented. Conversely, models like STAN or CSRM, which might be more robust to shorter sequences or different sparsity patterns, show better adaptability on Diginetica.

The varying performance across datasets underscores the importance of dataset-specific model tuning and the ongoing challenge of building highly generalizable sequential recommendation models. While **LSTM** excels in scenarios with richer sequential data (YOOCHOOSE), its performance on sparser or more diverse datasets like Diginetica suggests avenues for future research. This could involve incorporating adaptive mechanisms for handling varying session lengths and sparsity levels, exploring hybrid approaches that combine the strengths of different models to improve robustness across diverse datasets, or designing regularization techniques specifically tailored for sparse sequential inputs. Further analysis into the specific patterns learned by **LSTM** and other top-performing models on Diginetica could reveal key differences in their ability to handle different data distributions and sparsity levels.

In summary, the results affirm the strong capabilities of **LSTM** in capturing sequential dynamics for recommendation, particularly on dense transactional datasets. However, they also highlight the persistent challenges in achieving universal optimal performance across datasets with significantly different characteristics, emphasizing the need for continued research into model robustness and generalizability.

3.6 Conclusion

In this chapter, we introduced our session-based recommender system powered by Long Short-Term Memory (LSTM) networks to effectively capture short-term user behavior in e-commerce settings. Unlike traditional approaches that rely on long-term user histories, our model focuses on in-session interactions, making it especially effective in handling scenarios involving anonymous users.

Based on evaluation results, the model shows highly competitive performance across benchmark datasets, successfully learning sequential patterns to deliver timely and contextually relevant product recommendations. These findings underscore the strong potential of our model for real-time applications where dynamic personalization and responsiveness are essential, and in the next chapter we explore how our session-based recommendation model can be integrated into a full-scale e-commerce platform to power intelligent, adaptive, and engaging shopping experience.

Chapter 4

Building an E-Commerce Platform Powered by Session-Based Recommendations

4.1 Introduction

After developing and evaluating our session-based recommender system using LSTM, the next logical step is to bring the model into a real-world setting where its benefits can be directly experienced by users. To achieve that, we developed a full-featured e-commerce platform that serves as a deployment environment for the session-based recommendation system. This chapter details the implementation of the platform and the integration of the recommendation system within it, enabling real-time, personalized product suggestions. By continuously responding to user behavior during each session, the system delivers a more dynamic, context-aware, and engaging shopping experience—even in the absence of historical user data.

4.2 E-Commerce Platform Architecture

Building an e-commerce platform powered by session-based recommendations requires a well-orchestrated architecture that bridges real-time user interactions with intelligent product suggestions.

At the frontend, users engage with an intuitive and responsive web interface designed to ensure seamless navigation and a consistent experience across devices.

Behind the interface, a robust backend system manages user sessions, processes client requests via a dedicated API layer, and coordinates with the data layer. This architecture facilitates seamless communication between the frontend and backend, ensuring timely access to essential resources such as product catalogs, user profiles, and session histories (see Figure 4.1).

At the core of the system lies the deep learning component—a trained session-based recommendation model that interprets user interactions occurring within individual browsing sessions to generate meaningful shopping insights. As users engage with a sequence of products—viewing items, adding them to the cart, or navigating through various categories—the system continuously captures and processes these behavioral patterns. This session-level data forms the foundation for real-time, context-aware recommendations.

This real-time session data serves as the input to the recommendation engine, which identifies sequential patterns and infers short-term user intent. By comparing ongoing sessions to learned behavioral trends, the model produces timely and contextually relevant product suggestions.

A continuous feedback loop further enhances the system's performance: each interaction not only triggers immediate recommendations but also adds fresh session data used to fine-tune the LSTM model. By periodically retraining on these new interactions, the platform adapts dynamically, delivering increasingly accurate and personalized experiences—even for users without long-term histories.

In summary, the platform transforms passive browsing into active discovery, leveraging session-based deep learning to intelligently connect users with products they are likely to engage with—ultimately driving both customer satisfaction and business outcomes.

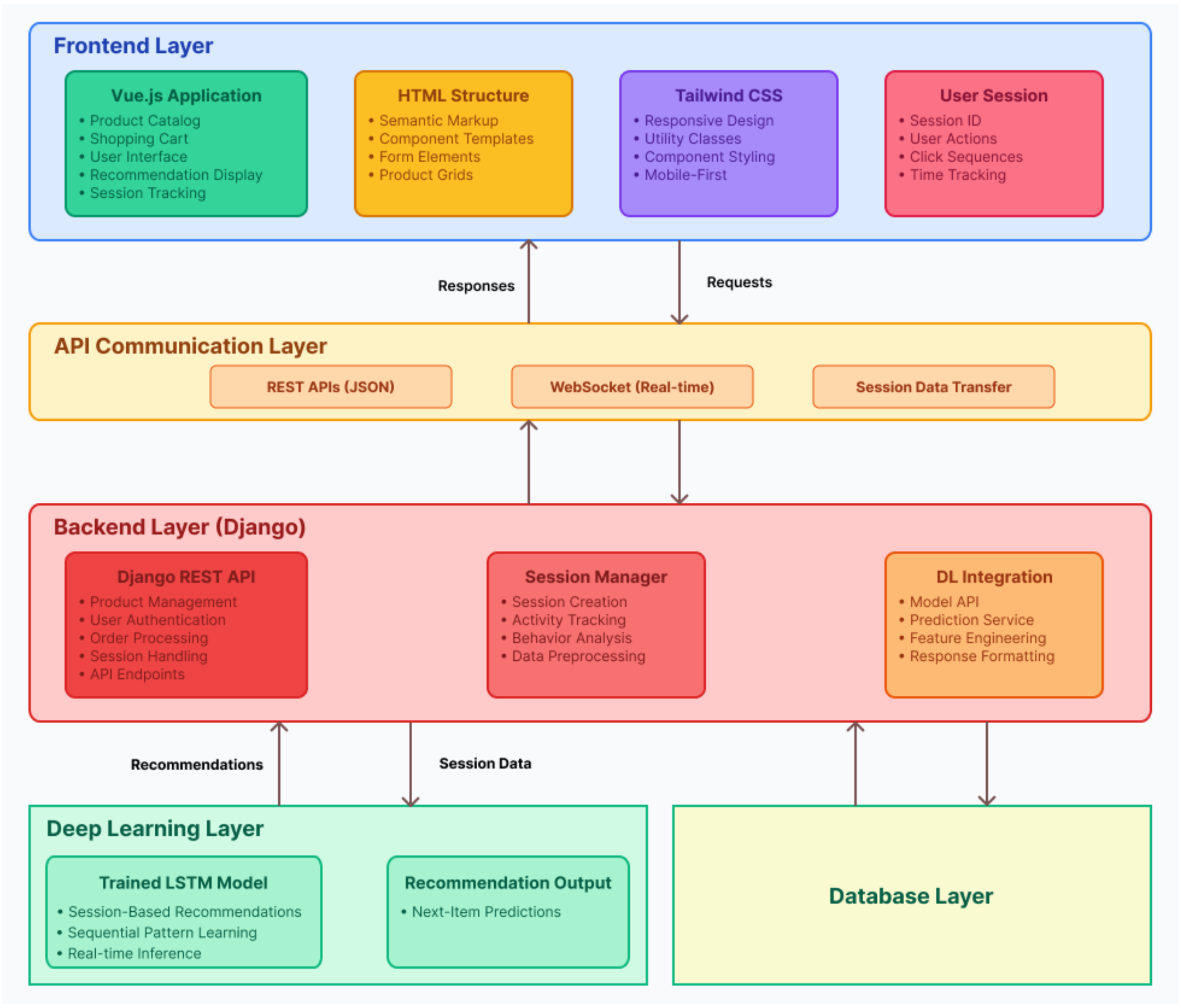


Figure 4.1: Architecture of the E-Commerce Platform with Session-Based Recommendations

4.3 Analysis and Conception

Following the overview of the system architecture, this section delves into a detailed analysis and conceptual modeling of the e-commerce platform. It provides a thorough examination of the platform's core requirements and design principles, focusing on defining its functional scope, structural components, and key behavioral scenarios to establish a solid foundation for implementation

4.3.1 Functional Description of the E-Commerce Platform

The developed platform offers a comprehensive e-commerce experience, supporting a wide range of operations tailored to the needs of customers, vendors, and administrators. Designed for modularity, scalability, and responsiveness, the system ensures an intuitive interface for end-users and efficient control tools for backend users. The platform also incorporates real-time, session-based recommendation capabilities that enhance personalization and user engagement.

User Role	Functionalities
Buyer	• Browse products by category and filters • Search for items using keywords • View product detail pages • Add items to cart and place orders • Register, log in/out, and manage profile • Submit product reviews • Communicate with vendors via messaging • Receive real-time personalized product recommendations based on current session behavior—even when browsing anonymously
Seller	• Create, update, and manage product listings • Edit store/profile information • View and respond to customer messages • Monitor orders related to their products
Administrator	• Manage platform users (customers and vendors) • Review and moderate vendor-submitted products • Manage product categories and subcategories • Oversee product reviews and order records • Ensure overall platform integrity and policy compliance

Table 4.1: User Roles and Functionalities

4.3.2 UML Diagrams Overview

After outlining the functional aspects of the platform in the previous subsection and summarizing them in Table 4.1, this part presents the conceptual modeling of the system using standard Unified Modeling Language (UML) diagrams. These diagrams help clarify the components of the system and how they interact during different operations.

Three key types of UML diagrams are used:

- **Use Case Diagram**: Represents actors and their interactions with system functions from a user's perspective.
- **Class Diagram**: Represents the system's structure: classes, their attributes, methods, and relationships between them.
- **Sequence Diagram**: Represents how objects interact over time: the step-by-step message flow for a specific scenario.

These diagrams provide a comprehensive and multidimensional view of the platform's design, bridging high-level architecture and specific implementation details.

Use Case Diagram

To visually represent the interactions between different user roles and the system, the following use case diagram (Figure 4.2) illustrates the main functionalities available to customers, vendors, and administrators. This diagram highlights the core system operations from the perspective of each actor, providing a high-level overview of the system's functional requirements as outlined in the initial functional description (Table 4.1). It serves as a bridge between the textual description of features and the detailed design phase, ensuring clarity in understanding user-system interactions.

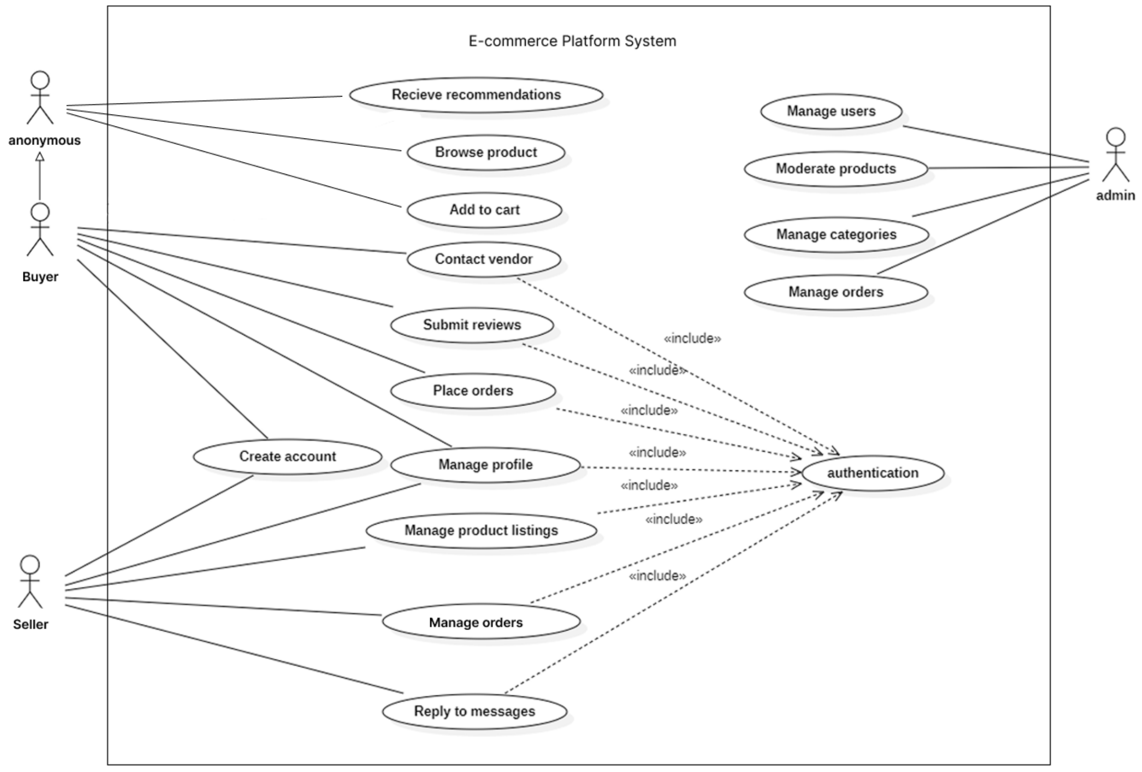


Figure 4.2: Use Case Diagram illustrating core system functionalities

Class Diagram

The class diagram presented in Figure 4.3 provides a detailed view of the static structure of the e-commerce platform, illustrating the fundamental classes, their attributes, operations, and the relationships between them. This diagram serves as a blueprint for the system's data model and object-oriented design, capturing the core entities and their interactions within the platform's domain.

- **User and Roles:** The system revolves around the `User` entity, specialized into `Buyer`, `Seller`, and `Admin` roles. Each role inherits basic user properties while possessing distinct permissions and functionalities relevant to their interaction with the platform, such as product management for sellers or platform oversight for administrators.
- **Product Catalog:** The `Product` class represents items available for sale, holding details like name, description, and price. Products are organized hierarchically through `Category` and `Subcategory` classes, enabling structured browsing and management.
- **Transaction Management:** The core e-commerce flow is supported by classes managing the shopping and ordering process. The `Cart` and `CartItem` classes handle temporary storage of products selected by a buyer. Upon checkout, an `Order` is created, containing `OrderItem` details that capture the specifics of the purchased products. Buyers can also provide feedback through the `Review` class, linked to both the buyer and the product.
- **Session-Based Recommendations:** Key components enabling personalization include `UserSession`, which tracks browsing activity (even for anonymous users), and `SessionAction`, which records specific interactions such as views or searches. The `RecommendationEngine` utilizes this session data to generate tailored product suggestions, demonstrating a dependency on session activity.
- **Communication:** Basic user interaction is facilitated by the `ChatMessage` class, enabling messaging between platform users such as buyers and sellers.

In essence, this class diagram provides a structured overview of the system's components and their interconnections, clarifying how user roles, products, transactions, and the recommendation system are integrated at the data model level.

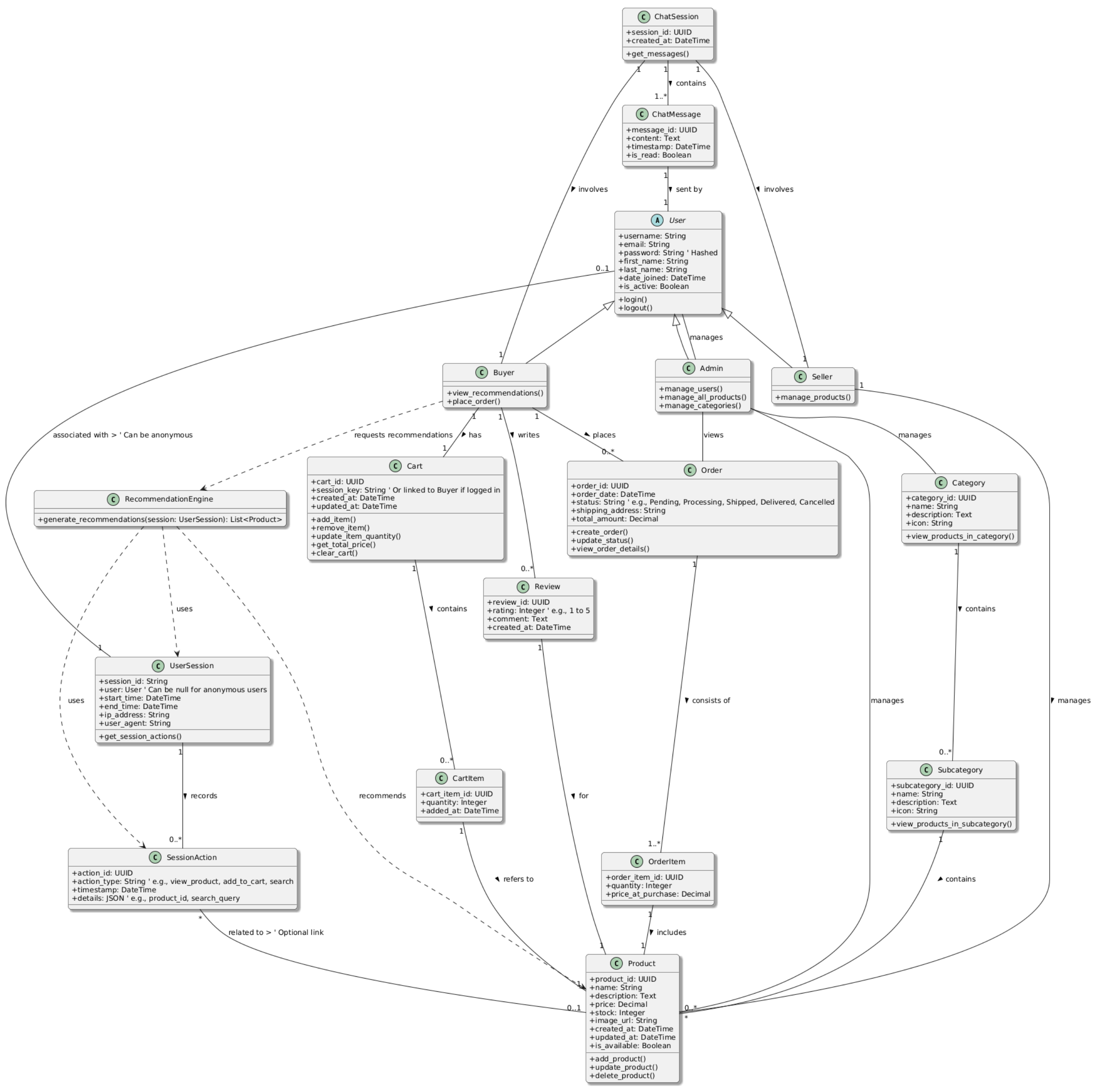


Figure 4.3: Class Diagram illustrating the static structure of the platform

Sequence Diagrams

While class diagrams illustrate the static structure, sequence diagrams model the dynamic behavior of the system by detailing the interactions between objects over time for specific scenarios. They are invaluable for understanding how different components collaborate to fulfill user requests and execute core processes.

Two critical sequences are presented here: placing an order (Figure 4.4) and generating session-based recommendations (Figure 4.5).

Placing an Order Scenario

This sequence details the process initiated when a buyer clicks 'Place Order'. The browser sends the request to the web server, which coordinates with the cart service to get item details and the order service to create the order. The order service checks stock availability in the database. If stock is sufficient, the order is created, stock is updated, the cart is cleared, and the user is redirected to a success page. If stock is insufficient, an error is shown, and the order is not placed.

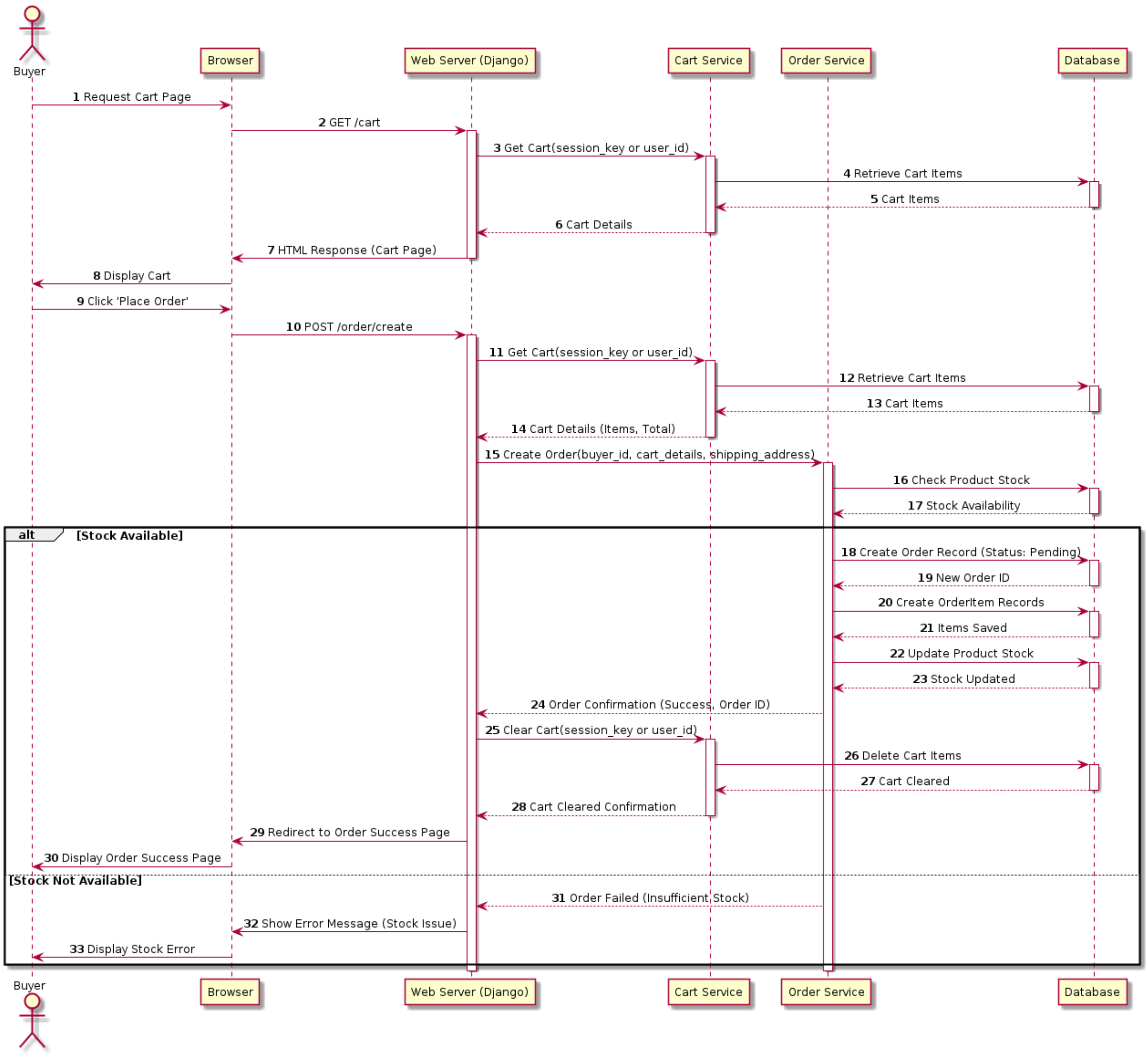


Figure 4.4: Sequence Diagram illustrating the process of placing an order

Generating Session-Based Recommendations Scenario

This sequence starts when a user views a product. The browser requests the page from the web server, which interacts with the session manager to log the user's action (e.g., product view) and retrieve the current session. The web server then asks the recommendation engine for suggestions based on the session's activity. The engine analyzes the session actions (fetched via the session manager from the database), identifies relevant products, gets their details, and returns them to the web server. The server integrates these recommendations into the page sent back to the user's browser.

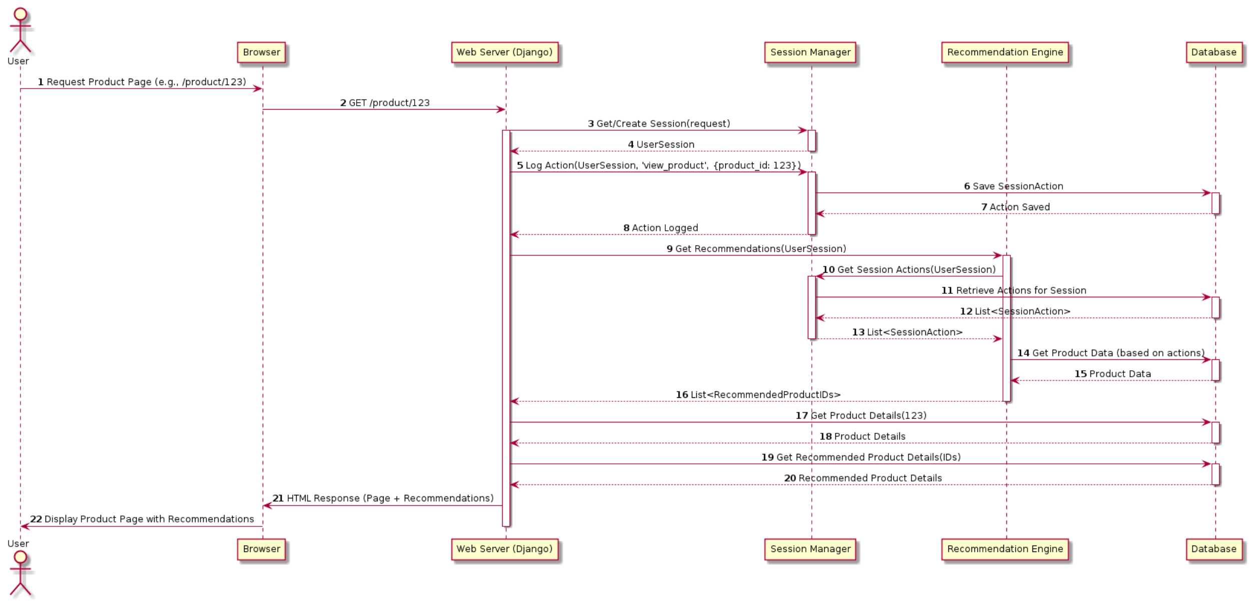

User
Browser
Web Server (Django)
Session Manager
Recommendation Engine
Database
1 Request Product Page (e.g., /product/123)
2 GET /product/123
3 Get/Create Session(request)
4 UserSession
5 Log Action(UserSession, 'view_product', {product_id: 123})
6 Save SessionAction
7 Action Saved
8 Action Logged
9 Get Recommendations(UserSession)
10 Get Session Actions(UserSession)
11 Retrieve Actions for Session
12 List<SessionAction>
13 List<SessionAction>
14 Get Product Data (based on actions)
15 Product Data
16 List<RecommendedProductIDs>
17 Get Product Details(123)
18 Product Details
19 Get Recommended Product Details(IDs)
20 Recommended Product Details
21 HTML Response (Page + Recommendations)
22 Display Product Page with Recommendations

Figure 4.5: Sequence Diagram illustrating the generation of session-based recommendations

4.4 E-Commerce Platform Implementation

This project was developed following a structured software engineering approach, starting with the analysis and design (conception) phases. After validating the design, we proceeded to the implementation phase using a modern and efficient technology stack suited for building responsive and interactive web applications. The technologies were selected to ensure scalability, maintainability, and a smooth user experience across devices.

4.4.1 Frontend Development Tools

HTML (HyperText Markup Language)

HTML[1] is the standard markup language for creating web pages and web applications. It provides the structure and content of a web page using a system of tags and elements, defining headings, paragraphs, images, links, and more, which web browsers then interpret and display.

Figure 4.6: HTML logo

[1]https://developer.mozilla.org/en-US/docs/Web/HTML

CSS (Cascading Style Sheets)

CSS[2] is a style sheet language used for describing the presentation of a document written in HTML (or XML). It allows developers to control the layout, colors, fonts, and overall visual appearance of web pages, separating content from presentation.

Figure 4.7: CSS logo

JavaScript

JavaScript[3] is a high-level, interpreted programming language primarily used to make web pages interactive. It enables complex features on web pages, such as interactive maps, animated graphics, and dynamic content updates, and is an essential technology alongside HTML and CSS for building modern web applications.

Figure 4.8: JavaScript logo

Tailwind CSS

Tailwind CSS[4] is a utility-first CSS framework for rapidly building custom user interfaces. Instead of predefined components, it provides low-level utility classes that can be composed directly in HTML to style elements, allowing developers to create highly customized designs without writing custom CSS.

Figure 4.9: Tailwind CSS logo

[2] `https://developer.mozilla.org/en-US/docs/Web/CSS`
[3] `https://developer.mozilla.org/en-US/docs/Web/JavaScript`
[4] `https://tailwindcss.com`

Vue.js

Vue.js[5] is an open-source progressive JavaScript framework for building user interfaces and single-page applications. It is designed to be incrementally adoptable, meaning it can be integrated into projects piece by piece, and is known for its approachability, performance, and versatility.

Figure 4.10: Vue.js logo

4.4.2 Backend Development Tools

Django

Django[6] is a high-level Python web framework that streamlines the development of secure and scalable web applications. It follows the Don't Repeat Yourself (DRY) principle to minimize code duplication and promote maintainability. Django provides an integrated set of components, including an object-relational mapper (ORM) for database interaction, URL routing, templating, and user authentication, enabling developers to build complex applications efficiently while adhering to best practices in software design.

Figure 4.11: Django logo

SQLite

SQLite[7] is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured SQL database engine. It is the most widely deployed database engine in the world, often used as an embedded database within applications rather than a standalone server, making it lightweight and easy to integrate.

[5]`https://vuejs.org`
[6]`https://www.djangoproject.com`
[7]`https://www.sqlite.org`

Figure 4.12: SQLite logo

4.4.3 Software Development and Design Tools

Visual Studio Code (VS Code)

Visual Studio Code (VS Code)[8] is a free, open-source code editor developed by Microsoft. It's highly extensible and popular for its rich features, built-in Git support, debugger, and a vast marketplace of extensions for various programming languages, frameworks, and development tools.

Figure 4.13: Visual Studio Code logo

StarUML

StarUML[9] is a sophisticated UML (Unified Modeling Language) diagramming tool used by software engineers and architects to model and document software systems. It supports various UML diagram types, such as class diagrams, sequence diagrams, and use case diagrams, aiding in system analysis and design.

Figure 4.14: StarUML logo

[8] `https://code.visualstudio.com/`
[9] `https://staruml.io/`

Figma

Figma[10] is a cloud-based design and prototyping tool for user interface (UI) and user experience (UX) design. It enables real-time collaboration, allowing multiple designers to work on the same file simultaneously, and is widely used for creating wireframes, mockups, and interactive prototypes for web and mobile applications.

Figure 4.15: Figma logo

GitHub

GitHub[11] is a web-based platform that uses Git for version control. It's a widely used platform for software development and collaboration, providing hosting for software projects, issue tracking, and features like pull requests for code review, making it central to open-source and commercial development workflows.

Figure 4.16: GitHub logo

4.5 Prototype

This section presents the prototype of the developed platform, showcasing the user interface and core features through a series of annotated screenshots. The prototype reflects the functionalities defined in the design phase and is built using the tools and technologies previously described. Emphasis is placed on usability, responsiveness, accessibility, and user experience across different devices and languages.

Responsive Design

The platform is fully responsive, providing an optimal user experience across devices. The layout dynamically adapts to different screen sizes, ensuring seamless navigation and interaction on both desktop and mobile interfaces.

[10] https://www.figma.com/
[11] https://github.com/

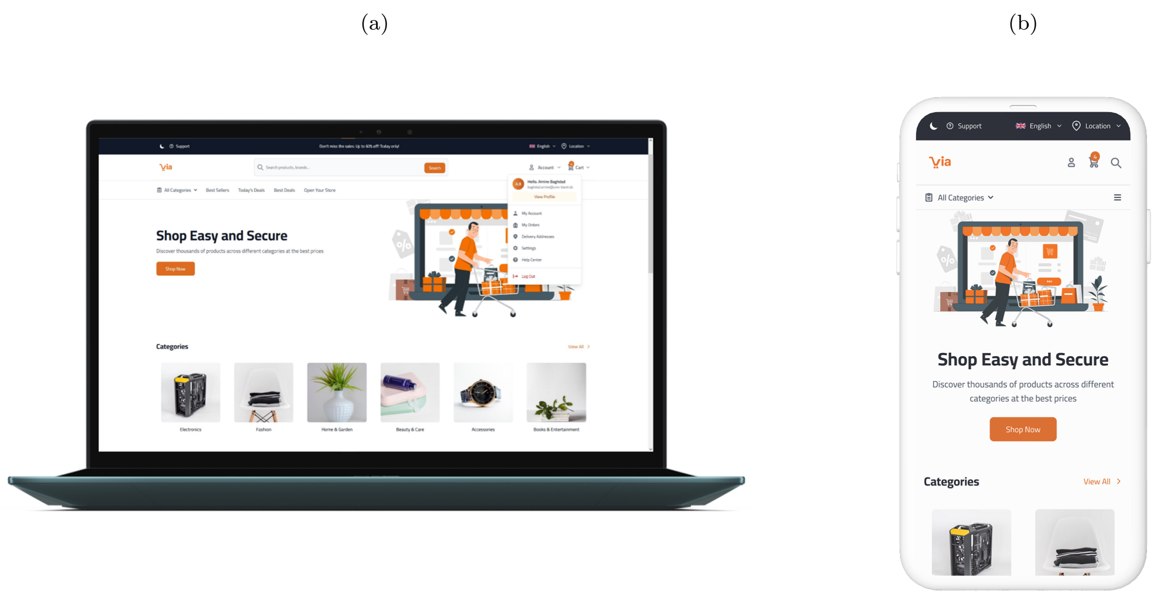


Figure 4.17: Responsive layout of the platform: (a) Desktop view, (b) Mobile view

Multilingual Support

To accommodate a diverse user base, the platform supports three languages: Arabic, French, and English. Users can easily switch between languages using a selector in the header, and the content updates accordingly.

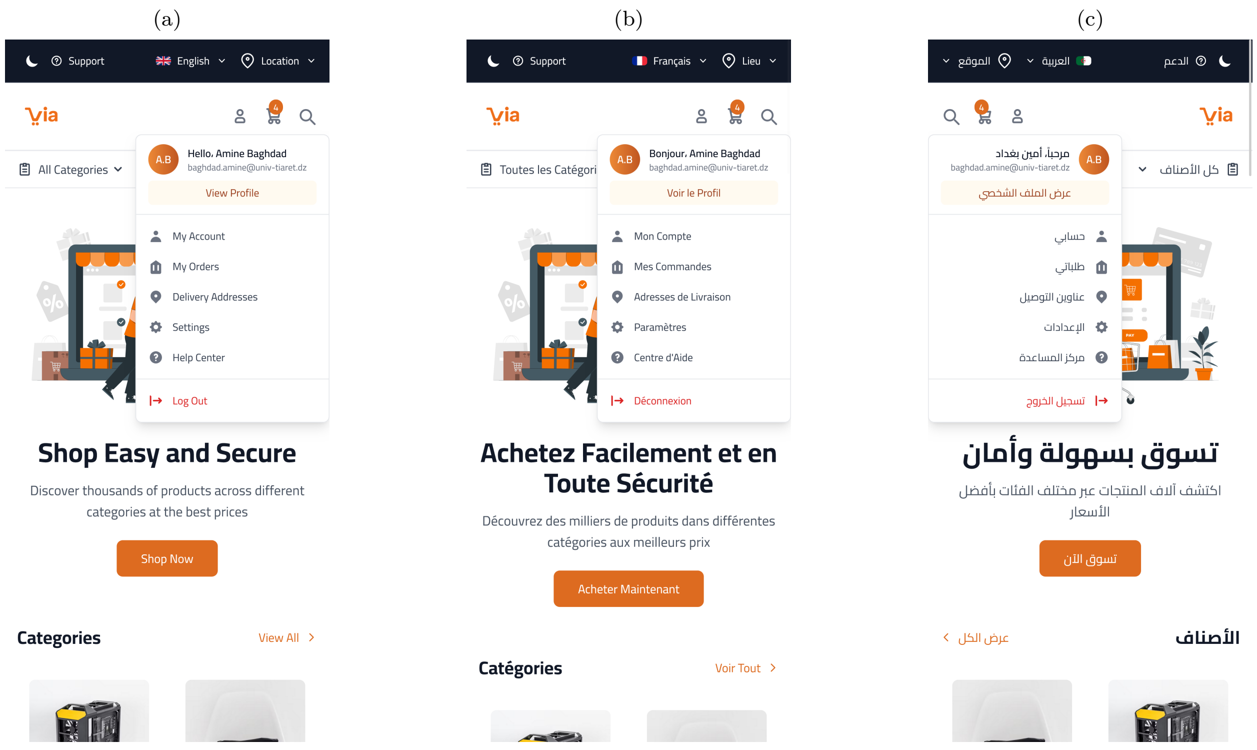


Figure 4.18: Multilingual interface support: (a) English, (b) French, (c) Arabic

Dark Mode

The application includes a dark mode toggle, enhancing accessibility and providing a more comfortable viewing experience in low-light environments.

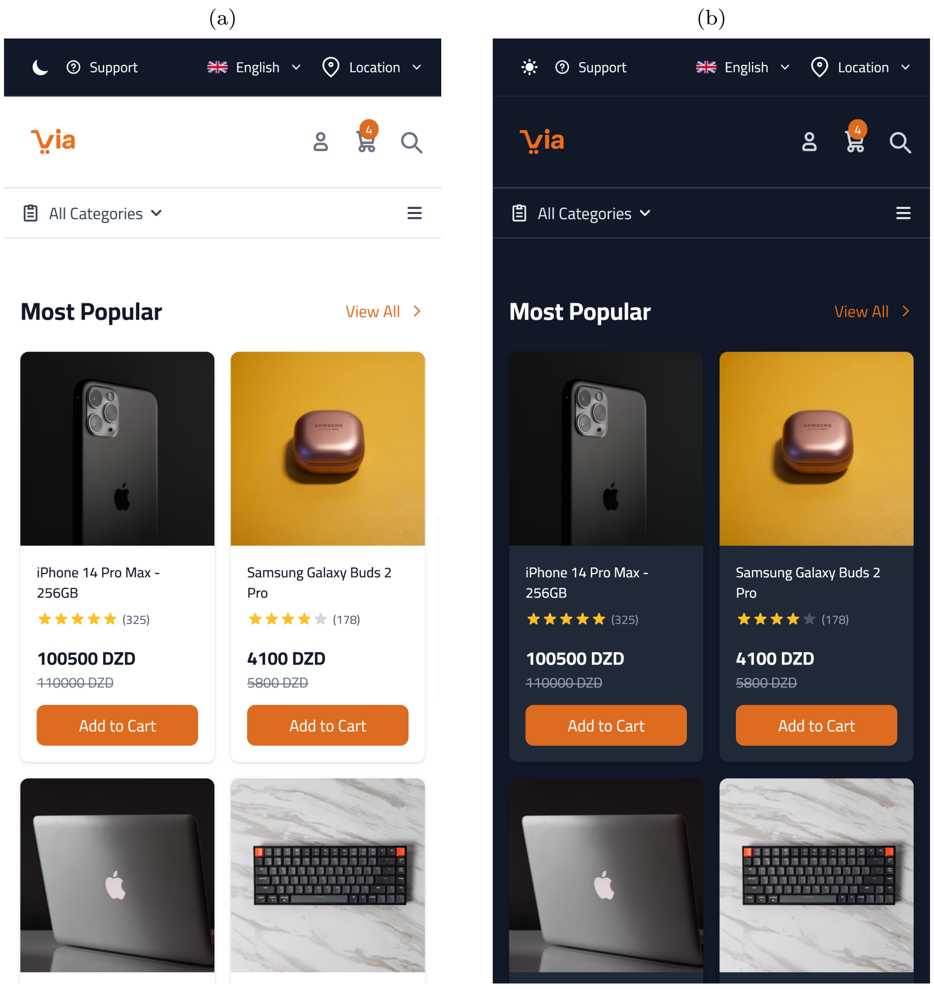


Figure 4.19: Interface appearance: (a) Light mode, (b) Dark mode.

Authentication System

Although the platform supports anonymous browsing and allows customers to add items to their cart without creating an account, a secure authentication system is required to confirm and place orders. This includes user registration, login, and OTP verification for enhanced security.

(a) (b)

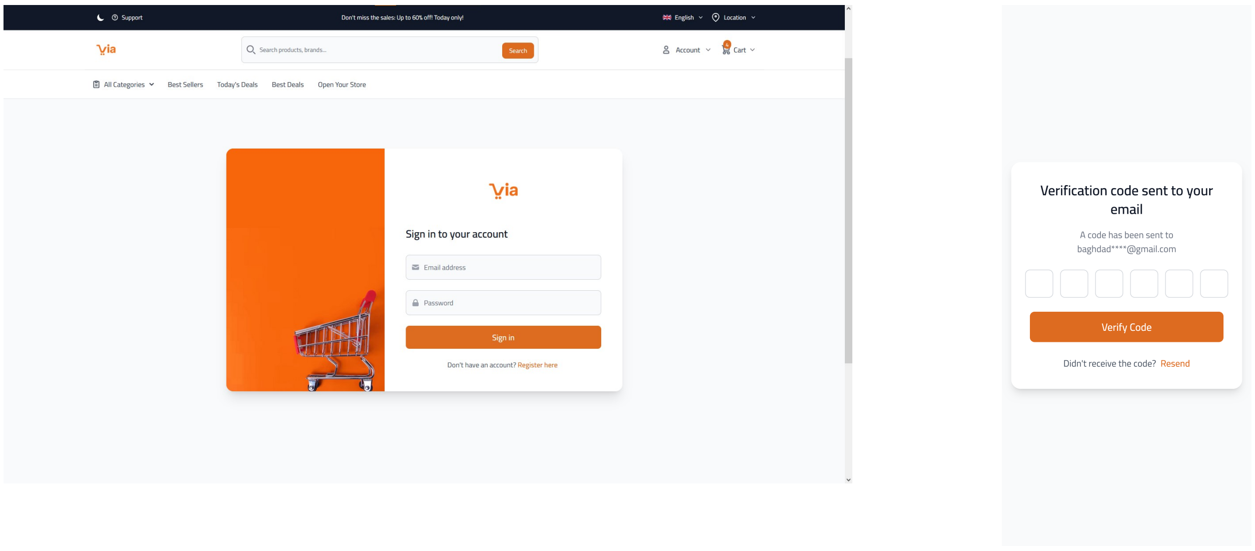
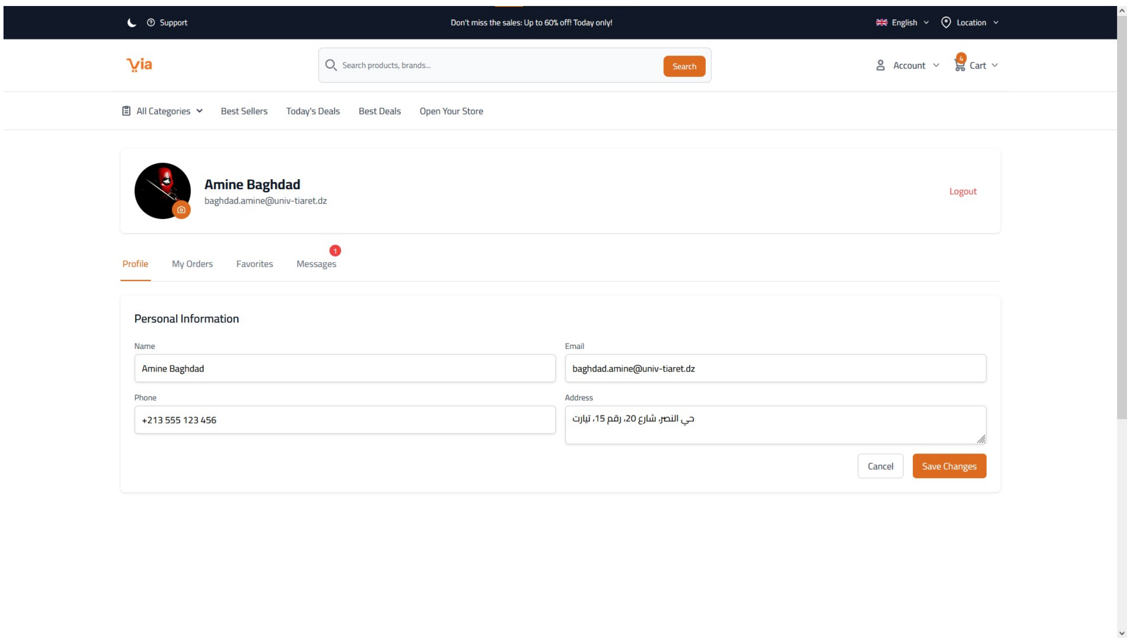

Support
Don't miss the sales: Up to 60% off! Today only!
English
Location
Via
Search products, brands...
Search
Account
Cart
All Categories
Best Sellers
Today's Deals
Best Deals
Open Your Store
Sign in to your account
Email address
Password
Sign in
Don't have an account? Register here
Verification code sent to your email
A code has been sent to baghdad****@gmail.com
Verify Code
Didn't receive the code? Resend

Figure 4.20: (a) Login screen and (b) OTP verification process

User Profile and Seller Dashboard

The platform provides distinct interfaces tailored to the roles of customers and sellers, enabling them to manage their respective activities efficiently.

- **User Profile**: The interface allows customers to update their personal information, track the status of their orders, manage favorite items, and communicate with sellers through the integrated messaging system. These functionalities are designed to enhance the overall user experience and provide transparency throughout the purchasing process.



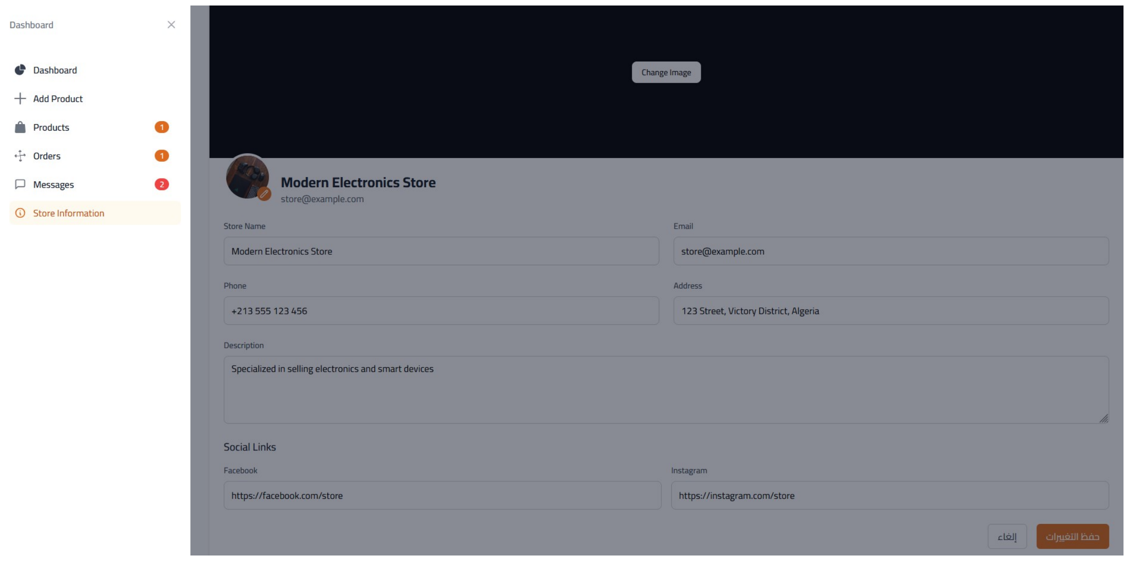
Amine Baghdad
baghdad.amine@univ-tiaret.dz

Personal Information

Name: Amine Baghdad
Email: baghdad.amine@univ-tiaret.dz
Phone: +213 555 123 456
Address: حي النصر، شارع 20، رقم 15، تيارت

Figure 4.21: User Profile Interface

- **Seller Dashboard**: Offers a centralized environment for sellers to operate their online stores. It includes features such as product management (adding, editing, and removing items), order handling (confirming, accepting, or rejecting orders), updating tracking information. Additionally, sellers can customize their store details and respond to customer messages directly through the platform.



Modern Electronics Store
store@example.com

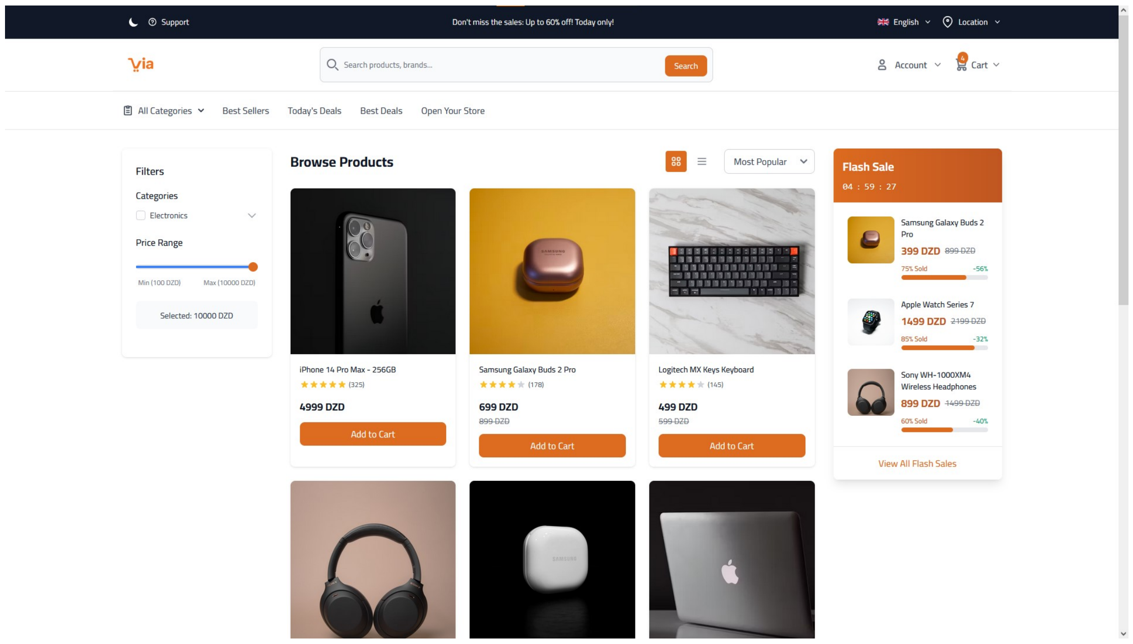
Store Name: Modern Electronics Store
Email: store@example.com
Phone: +213 555 123 456
Address: 123 Street, Victory District, Algeria
Description: Specialized in selling electronics and smart devices

Social Links
Facebook: https://facebook.com/store
Instagram: https://instagram.com/store

Figure 4.22: Seller Dashboard Interface

Product Browsing and Filter

Users can explore products through a structured catalog organized by categories and sub-categories. Filters such as price range and sorting options help users quickly find items that match their preferences.



iPhone 14 Pro Max - 256GB ★★★★★ (325) 4999 DZD

Samsung Galaxy Buds 2 Pro ★★★★ (178) 699 DZD ~~899 DZD~~

Logitech MX Keys Keyboard ★★★★ (145) 499 DZD ~~599 DZD~~

Samsung Galaxy Buds 2 Pro 399 DZD ~~899 DZD~~ 75% Sold -56%

Apple Watch Series 7 1499 DZD ~~2199 DZD~~ 85% Sold -32%

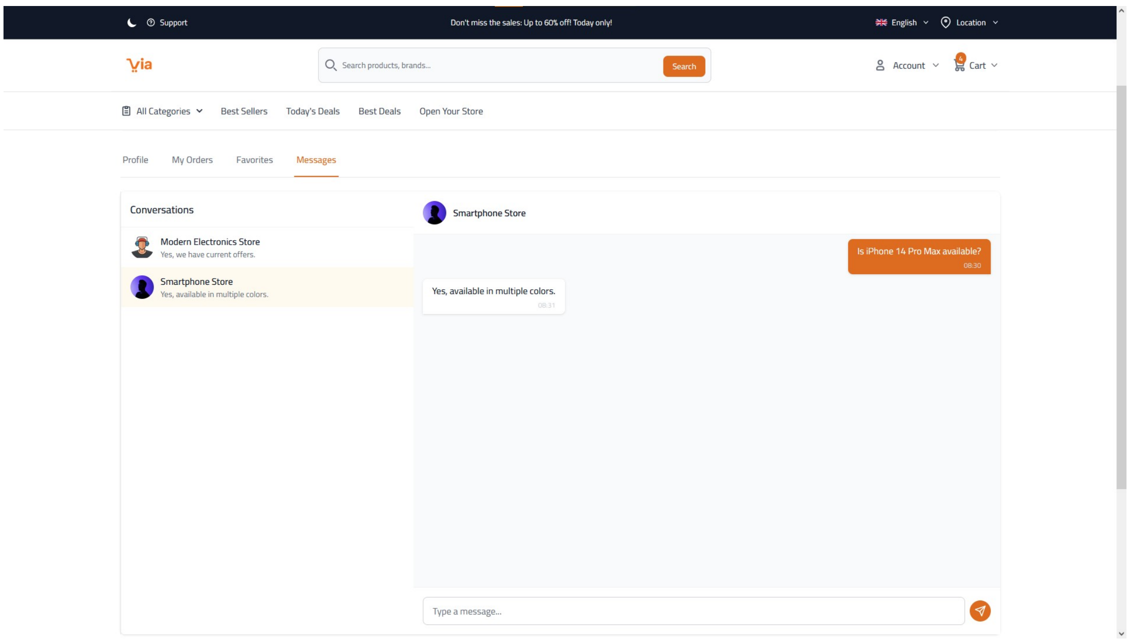
Sony WH-1000XM4 Wireless Headphones 899 DZD ~~1499 DZD~~ 60% Sold -40%

Figure 4.23: Product browsing with filter options

Real-Time Chat System

The platform integrates a real-time chat system that enables seamless communication between customers and sellers. This feature enhances customer support and allows users to ask questions, request product details, or negotiate before making a purchase.

The chat system updates messages instantly without requiring page reloads, providing a dynamic and responsive experience.



Modern Electronics Store
Yes, we have current offers.

Smartphone Store
Yes, available in multiple colors.

Smartphone Store

Is iPhone 14 Pro Max available? 08:30

Yes, available in multiple colors. 08:31

Figure 4.24: Real-time chat interface between a customer and a seller

Real-Time Session-Based Recommendation

The platform features a real-time recommendation system that adapts to users' current session behavior. As users interact with products, their actions are instantly analyzed to generate relevant suggestions. The process includes tracking user actions and displaying personalized recommendations in key interface areas.

- **User Actions**: The recommendation system captures session interactions in real time, including product views, and cart additions. This information is then processed by the backend model to understand user intent within the session.

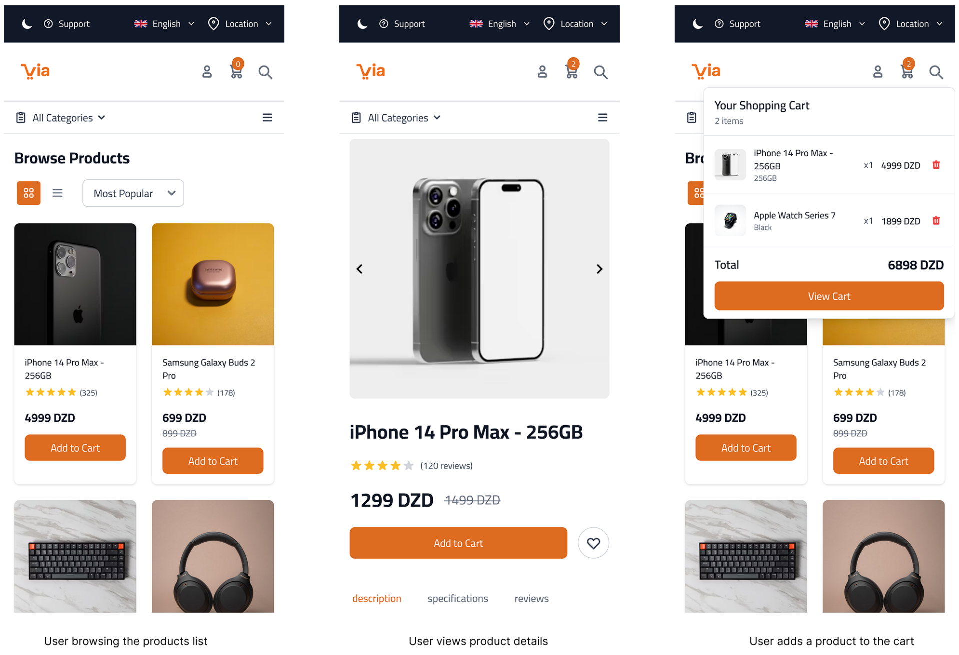


Figure 4.25: User behavioral actions tracked in real-time

- **Recommendations Placement**: The generated suggestions are displayed across key interface areas, such as the homepage, product detail pages, and search page. These placements are chosen to ensure maximum visibility and contextual relevance, increasing the chances of user engagement and conversion.

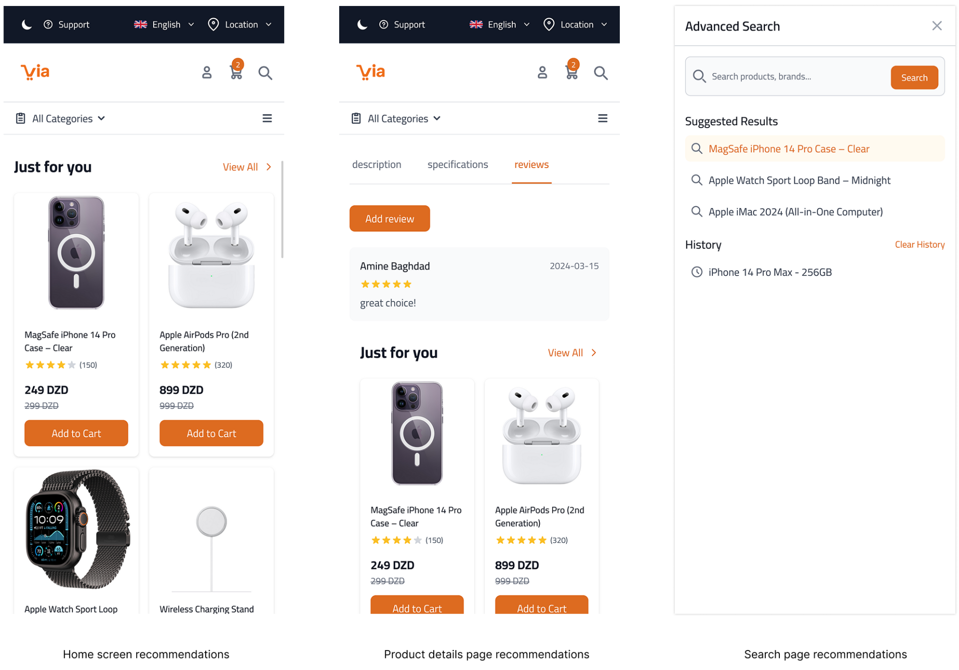

Just for you
View All
MagSafe iPhone 14 Pro Case – Clear
249 DZD
Apple AirPods Pro (2nd Generation)
899 DZD
Add to Cart
Apple Watch Sport Loop
Wireless Charging Stand
Add review
Amine Baghdad
2024-03-15
great choice!
Advanced Search
Search products, brands...
Search
Suggested Results
MagSafe iPhone 14 Pro Case – Clear
Apple Watch Sport Loop Band – Midnight
Apple iMac 2024 (All-in-One Computer)
History
Clear History
iPhone 14 Pro Max - 256GB
Home screen recommendations
Product details page recommendations
Search page recommendations

Figure 4.26: Real-time recommendations on homepage, product page, and search

4.6 Conclusion

In this chapter, we presented the design and implementation of a fully functional e-commerce platform integrating a session-based recommendation system powered by LSTM. By embedding the model within a real-time browsing environment, the platform dynamically tailors product suggestions based on users' current interactions. The system successfully demonstrates how session-based recommendations can enhance user experience without relying on prior user history. This practical application underscores the value of the model and sets the stage for further improvements.

General Conclusion

Summary

This thesis explored the application of Long Short-Term Memory (LSTM) networks for session-based recommendation systems (SBRSs), with a particular focus on enhancing next-item prediction within anonymous user sessions in the e-commerce domain.

The main research question guiding this work was:

> *"How can LSTM-based models be effectively applied to capture the sequential and evolving patterns of user behavior in session-based recommendation systems to enhance recommendation quality in e-commerce platforms?"*

To address this, the study was structured into four core chapters. The first chapter provided a comprehensive overview of recommender systems, emphasizing the unique challenges and opportunities associated with SBRSs, especially in environments where users interact without persistent identities. The second chapter introduced foundational concepts in Artificial Intelligence and Deep Learning, with a focus on Recurrent Neural Networks (RNNs), particularly LSTM networks, highlighting their relevance for modeling sequential behavior.

Chapter three described the design and implementation of an LSTM-based session recommender, detailing data preprocessing, sequence modeling, network architecture, and performance evaluation using standard metrics such as Hit Rate and Mean Reciprocal Rank (MRR). Experimental results on public datasets confirmed that LSTM networks are capable of learning complex sequential dependencies within sessions, producing accurate and timely item recommendations.

Finally, chapter four demonstrated the integration of the trained LSTM model into a functional e-commerce platform, bridging theory and practice. This chapter illustrated how the recommendation engine can power personalized shopping experiences in real-world scenarios.

The proposed LSTM-based model demonstrated promising results by effectively capturing temporal dependencies in item interaction sequences, balancing performance and simplicity without requiring long-term user histories or extensive contextual data. However, it has limitations, including sensitivity to hyperparameter tuning, reliance solely on item ID sequences without incorporating metadata or broader user behavior, and challenges in real-time scalability. Overall, LSTM architectures remain a strong choice for session-based recommendation systems, particularly in anonymous settings, due to their ability to retain temporal context. Future work should focus on improving robustness, scalability, and adaptability across varied scenarios.

Directions for Future Research

Based on the observed strengths and limitations of the LSTM-based model, several directions are proposed to guide future research:

- Integrating additional data sources such as item metadata, product categories, or real-time clickstream context to enrich session representations.

- Combining LSTMs with other models, including convolutional layers, attention mechanisms, or memory-augmented networks, to better capture both short-term and long-term dependencies.
- Exploring hybrid architectures that integrate LSTM-based SBRSs with collaborative filtering or content-based approaches for broader personalization capabilities.
- Evaluating robustness across domains, including cold-start scenarios, data sparsity, and behavioral drift, to ensure consistent performance in production systems.

Bibliography

[1] Geri Mileva. A helpful overview on ecommerce recommendation systems. `https://influencermarketinghub.com/ecommerce-recommendation-system/`, 2024. Accessed: 2025-04-14.

[2] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016.

[3] NVIDIA. What is a recommendation system? `https://www.nvidia.com/en-us/glossary/data-science/recommendation-system/`, n.d. Accessed: 2025-04-14.

[4] Zhenhua Dong and et al. A brief history of recommender systems. Accepted by DLP-KDD 2022, 2022. `https://dlp-kdd2022.github.io/`.

[5] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[6] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, 1995.

[7] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, 1995.

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[9] Joseph A Konstan, John Riedl, and Barry Smyth. Proceedings of the 2007 acm conference on recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, 2007.

[10] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: An end-to-end wide & deep learning framework for ctr prediction. *arXiv preprint arXiv:1804.04950*, 2018.

[12] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, New York, NY, USA, 2016. Association for Computing Machinery.

[13] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *Fourteenth ACM Conference on Recommender Systems*, pages 23–32, 2020.

[14] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 291–324. Springer, Berlin, Heidelberg, 2007.

[15] Hongde Zhou, Fei Xiong, and Hongshu Chen. A comprehensive survey of recommender systems based on deep learning. *Applied Sciences*, 13(20):11378, 2023.

[16] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9:59, 2022.

[17] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[18] Bihi Sabiri, Amal Khtira, Bouchra El Asri, and Maryem Rhanoui. Hybrid quality-based recommender systems: A systematic literature review. *Journal of Imaging*, 11(1):12, 2025.

[19] Andrew Schein, Alexandrin Popescul, Lyle Ungar, and David Pennock. Methods and metrics for cold-start recommendations. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, pages 253–260, 08 2002.

[20] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[21] Xiaoyuan Su and Taghi Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. Artificial Intelligence*, 2009, 10 2009.

[22] Gediminas Adomavicius and Jingjing Zhang. Stability of recommendation algorithms. *ACM Transactions on Information Systems*, 30:47–54, 09 2010.

[23] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. Internet Techn.*, 7, 2007.

[24] Sean McNee and John Riedl. Being accurate is not enough: How accuracy metrics have hurt recommender systems. *CHI '06 Extended Abstracts*, pages 1097–1101, 04 2006.

[25] Weiming Huang, Baisong Liu, and Hao Tang. Privacy protection for recommendation system: A survey. *Journal of Physics: Conference Series*, 1325:012087, 10 2019.

[26] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys*, 51, 02 2018.

[27] Jiahao Yuan, Wendi Ji, Dell Zhang, Jinwei Pan, and Xiaoling Wang. Micro-behavior encoding for session-based recommendation. In *Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2886–2899, 2022.

[28] Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28, 12 2018.

[29] Wikipedia contributors. Session (web analytics) — wikipedia, 2024. Accessed: 2025-05-01.

[30] Jinseok Jamie Seol, Youngrok Ko, and Sang-goo Lee. Exploiting session information in bert-based session-aware sequential recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 2639–2644. ACM, July 2022.

[31] Aaron Halfaker, Oliver Keyes, Daniel Kluver, Jacob Thebault-Spieker, Tien T. Nguyen, Kenneth Shores, Anuradha Uduwage, and Morten Warncke-Wang. User session identification based on strong regularities in inter-activity time. *CoRR*, abs/1411.2878, 2014.

[32] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):346–353, July 2019.

[33] Asia Biega, Peter Potash, Hal Daumé, Fernando Diaz, and Michèle Finck. Operationalizing the legal principle of data minimization for personalization. pages 399–408, 07 2020.

[34] Itishree Mohallick, Katrien De Moor, Özlem Özgöbek, and Jon Gulla. *Towards New Privacy Regulations in Europe: Users' Privacy Perception in Recommender Systems: 11th International Conference and Satellite Workshops, SpaCCS 2018, Melbourne, NSW, Australia, December 11-13, 2018, Proceedings*, pages 319–330. 12 2018.

[35] Wikipedia contributors. General data protection regulation, 2024. Accessed: 2025-05-01.

[36] Wikipedia contributors. California consumer privacy act, 2024. Accessed: 2025-05-01.

[37] Wikipedia contributors. Transport layer security, 2024. Accessed: 2025-05-01.

[38] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. *User Modeling and User-Adapted Interaction*, 27, 12 2017.

[39] Blendee. Recommandation de produits pour augmenter les ventes en ligne.

[40] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet Orgun, and Defu Lian. A survey on session-based recommender systems, 2021.

[41] LinkedIn. What are the main challenges and risks of implementing recommendation engines for ecommerce?, 2025. Accessed: 2025-05-11.

[42] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence: August 31, 1955. *AI Mag.*, 27(4):12–14, December 2006.

[43] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

[44] IBM. What is machine learning (ml)?, 2021. Accessed: 2025-05-02.

[45] Google for Developers. Machine learning glossary, 2025. Accessed: 2025-05-02.

[46] AltexSoft Editorial Team. Semi-supervised learning, explained with examples, 2024. Accessed: 2025-05-02.

[47] TutorialsPoint. Machine learning - limitations. `https://www.tutorialspoint.com/machine_learning/machine_learning_limitations.htm`. Accessed: 2025-05-02.

[48] Hyun Suk Lee and Junga Lee. Applying artificial intelligence in physical education and future perspectives. *Sustainability*, 13(1), 2021. Figure 2: Relationship between AI, machine learning, and deep learning.

[49] Wikipedia contributors. Deep learning — wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Deep_learning`. Accessed: 2025-05-03.

[50] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[51] Michael Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. `http://neuralnetworksanddeeplearning.com/`.

[52] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[53] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision (ICCV 2015)*, 1502, 02 2015.

[55] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[56] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[57] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[58] Fakultit Informatik, Y. Bengio, Paolo Frasconi, and Jfirgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*, 03 2003.

[59] Domor Mienye, Theo Swart, and George Obaido. Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information*, 15:517, 08 2024.

[60] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity, 2020.

[61] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997.

[62] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[63] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, 07 2017.

[64] Star: A session-based time-aware recommender system. *Neurocomputing*, 573:127104, 2024.

[65] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. *Proceedings of ACM World Wide Web Conference*, 1, 08 2001.

[66] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback, 2012.

[67] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. pages 811–820, 04 2010.

[68] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, and Jun Ma. Neural attentive session-based recommendation, 2017.

[69] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. Stamp: Short-term attention/memory priority model for session-based recommendation. pages 1831–1839, 07 2018.

[70] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. Sequence and time aware neighborhood for session-based recommendations: Stan. page 1069–1072, New York, NY, USA, 2019. Association for Computing Machinery.

[71] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten Rijke. A collaborative session-based recommendation approach with parallel memory modules. pages 345–354, 07 2019.

Appendices

Appendix 1: Business Model Canvas

TechnoFoster

Presentation 2025

IBN KHALDOUN
UNIVERSITY

نموذج مخطط الأعمال للمشاريع الخاصة بالقرار 1275

Content

Content

البطاقة التقنية للمشروع

Baghdad Amine	الاسم و اللقب Votre prénom et nom Your first and last Name
Via	السم التجاري للمشروع Intitulé de votre projet Title of your Project
(SARL – Société à Responsabilité Limitée)	الصفة القانونية للمشروع Votre statut juridique Your legal status
0783193758 0659437806	رقم الهاتف Votre numéro de téléphone Your phone number
baghdad.amine@univ-tiaret.dz baghdad.amine.div@gmail.com	البريد الالكتروني Votre adresse e-mail Your email address
Tiaret - Sougueur	مقر مزاولة النشاط (الولاية -البلدية) Votre ville ou commune d'activité Your city or municipality of activity

البطاقة التقنية للمشروع

.Via is a smart and modern Algerian e-commerce platform that offers a complete digital shopping experience, connecting customers with local merchants. The platform leverages artificial intelligence to deliver real-time, personalized product recommendations based on user interactions, enhancing the overall shopping journey. Via provides an intuitive interface and integrated services, including product browsing, order management, electronic payment, and delivery — making it an all-in-one solution for online buying and selling in Algeria.	.

القيمة المقترحة أو العرض المقدّم
Value Proposition

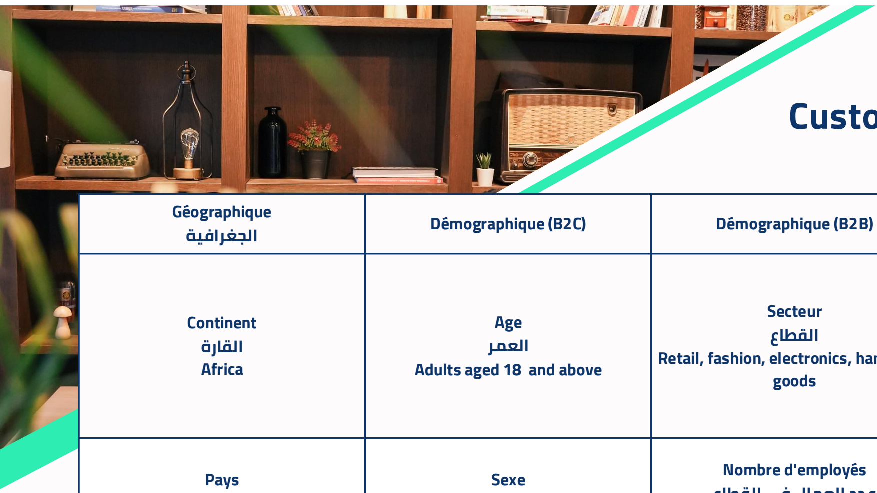
تحديد المشكل الذي يواجهه الزبون

E-commerce in Algeria lacks a reliable and intelligent platform. Most transactions still rely on ads and cash on delivery, causing high cancellation rates and seller losses. Prepayments expose buyers to scams, and delivery remains expensive and unclear. Via addresses these problems with a smart, secure platform offering integrated electronic payments, AI-driven recommendations, protected transactions, and improved delivery — making online shopping safer, smarter, and more efficient. .	ما هي المشكلة التي تريد حلها؟
There is no fully integrated and trusted Algerian e-commerce platform that provides an engaging user experience connecting merchants and customers. The most recognizable platforms mainly function as announcement or advertising sites where transactions occur outside the platform. Additionally, much of the e-commerce activity relies on Facebook Marketplace or Instagram, or simple websites that only accept orders without offering integrated payment systems or a complete e-commerce ecosystem, including recommendation systems, chat, and secure payments.	ما هي البيانات المتوفرة لديك التي تدل على وجود المشكلة المحددة؟
Projects like Jumia, Tidjara.dz, and Yassir have been launched to support online commerce and improve digital shopping experiences in Algeria.	ما هي المشاريع الأخرى التي استهدفت نفس المشكلة والتي جرى تنفيذها؟
The goal is to build a complete and intelligent e-commerce platform based in Algeria that offers a secure, modern, and fully integrated ecosystem. The platform aims to support local merchants and enhance the customer experience. It will provide a trustworthy environment where buyers and sellers can interact seamlessly, driving the growth of digital commerce in Algeria.	ماهي أهداف مشروعك و/أو نتائجه المتوقعة؟

القيمة المقترحة أو العرض المقدّم
Value Proposition

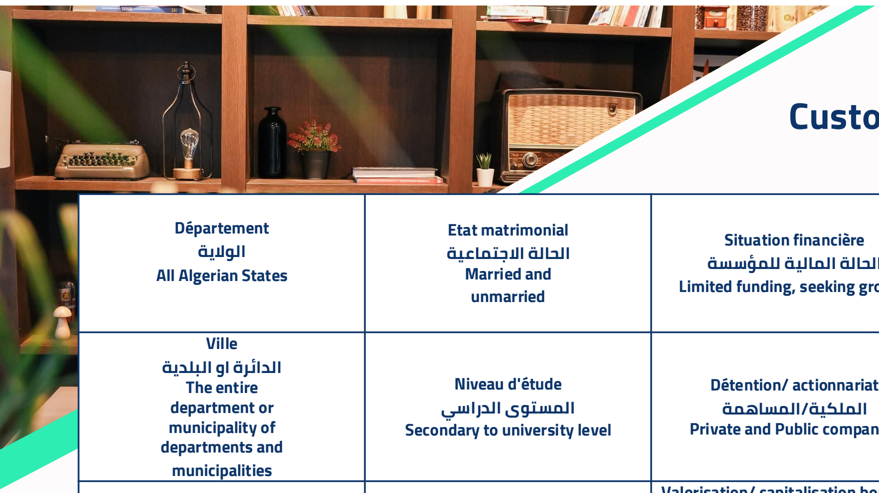
القيمة المقترحة وفق المعايير التالية

An intelligent Algerian e-commerce platform that leverages AI—especially session-based recommendations—to deliver a personalized, secure, and seamless shopping experience.	القيمة المبتكرة أو الجديدة
AI-driven personalization tailors product recommendations and the user experience to individual preferences and behavior.	القيمة بالتخصيص
Competitive and transparent pricing with attractive promotions tailored to the Algerian market.	القيمة بالسعر
Modern, intuitive, and user-friendly interface designed for smooth navigation and accessibility.	القيمة بالتصميم
An AI-powered platform that ensures fast browsing, accurate recommendations, and efficient order processing for a smooth and reliable shopping experience.	القيمة بالأداء العالي
A complete ecosystem from product discovery to payment, chat, and delivery — all in one place — with built-in protection for both buyers and sellers.	القيمة بالخدمة الشاملة
Support for local merchants, real-time tracking, and a smart AI core that evolves with user data to continuously enhance the experience	قيم أخرى



Customer Segmentsشرائح العملاء أو الزبائن

Géographique الجغرافية	Démographique (B2C)	Démographique (B2B)	Psychographique العوامل النفسية و الشخصية	Comportemental السلوكيات
Continent القارة Africa	Age العمر Adults aged 18 and above	Secteur القطاع Retail, fashion, electronics, handmade goods	Classe sociale لطبقة الاجتماعية General class middle class upper class	Usage استخدام Users can browse products, receive personalized recommendations, communicate with sellers via chat, place secure online orders, and track all within an integrated and —delivery .user-friendly environment
Pays الدولة Algeria	Sexe الجنس Both genders	Nombre d'employés عدد العمال في القطاع Small & Medium Enterprises	Niveau de vie المستوى المعيشي Medium to high standard of living	Loyauté الوفاء Customers prefer to continue using the platform for trusted service, and .regular updates
Région الجهة the entire national territory of Algeria	Revenus annuel متوسط الدخل Low, Middle, High income	Maturité de l'entreprise نضج المؤسسة A newly launched project in its early stages. Focused on building a strong and scalable platform. Limited financial resources with a small, dynamic team. Efforts are directed toward improving internal processes, structuring operations, and expanding market presence while managing costs efficiently	Valeurs القيم Trust, transparency, digital innovation, support for local businesses, and commitment to user satisfaction.	Intérêt اهتمام Interested in online shopping , promotions, convenience, delivery



Customer Segmentsشرائح العملاء أو الزبائن

Département الولاية All Algerian States	Etat matrimonial الحالة الاجتماعية Married and unmarried	Situation financière الحالة المالية للمؤسسة Limited funding, seeking growth	Personnalité الشخصية Seeks efficient and trustworthy e-commerce experiences	Passion الهواية و شغف People passionate about online shopping, discovering new products, and enjoying the convenience of digital .commerce
Ville الدائرة او البلدية The entire department or municipality of departments and municipalities	Niveau d'étude المستوى الدراسي Secondary to university level	Détention/ actionnariat الملكية/المساهمة Private and Public companies	Convictions المعتقدات Delivering quality products and services that respect religious, social, and cultural beliefs	Sensibilité حساسيات People are sensitive to product price and quality, delivery reliability, data .privacy, and transparency in service
Quartier الحي All Algerian neighborhoods	Profession المهنة All kinds of professions from all fields	Valorisation/ capitalisation boursière التقييم / القيمة السوقية The goal is to grow the platform's user base and build trust. Revenue will come from monthly subscriptions, service fees, and sponsorships. Future value will be driven by integrating AI to enhance marketing, personalization, and overall platform efficiency.	Présence digitale et sur les réseaux sociaux استعمال التكنولوجيا في التواصل Facebook, Instagram, occasional TikTok	Habitude de consommation عادة الاستهلاك Users frequently browse, compare, and buy products online, especially during .promotions or seasonal sales

Customer Segmentsشرائح العملاء أو الزبائن

Climat المناخ All Algerian Climat zone	Culture الثقافة Digitally-aware consumers with a growing culture of online shopping and openness to secure e-commerce experiences.	Business model نموذج الأعمال Revenue will be generated through individual subscriptions that allow users to access and promote their products or services, as well as corporate subscriptions that enable businesses to showcase their offerings to a wider audience. Additional income will come from service fees on transactions and strategic sponsorships.	Centres d'intérêts مراكز الاهتمام Shopping, fashion, home items, electronics, beauty products, gadgets, and discovering new deals online.	Mode de paiement طرق الدفع Payment can be made electronically or in cash upon delivery
	Religion الدين Aligned with Islamic values by offering halal products and ensuring ethical transactions.	Secteur servi القطاع الذي يخدمه E-commerce and retail, with dedicated support for local merchants, logistics services, and electronic payments.		Connaissance المعرفة Clients generally familiar with using mobile apps and websites for online shopping, browsing, and digital payments
	Langue اللغة Arabic, French , English	Technologie utilisée التكنولوجيا المستعملة Website and application laicifitrA ,(erutuf) Intelligence		Nature de la demande طبيعة الطلب Customers demand products either frequently, periodically, or occasionally based on their specific needs.
		Format du produit ou packaging شكل المنتج أو التعبئة والتغليف digital platform		Fréquence d'achat عدد مرات الطلب على السلعة Frequent and monthly purchase reminders and updates based on product availability and promotions

قنوات التوزيع
CHANNELS

Direct sales through the platform	المبيعات المباشرة
/	تجار الجملة
Official distributors or intermediaries selling specific brands/products via the platform	الموزعون
Selling directly to end consumers via the platform	توزيع التجزئة

Customer Relationship العلاقة مع العملاء

Multiple support channels and a responsive team to quickly resolve issues. Clear policies and encouraging reviews to enhance credibility. Using customer feedback to constantly improve the platform and services. Using AI to understand customers and provide tailored recommendations.	كيف تدير علاقاتك مع العملاء؟
Zoho CRM or HubSpot CRM	ماهية أهم البرامج التي ستعتمد عليها في ادارة العلاقة مع الزبون Microsoft Dynamics Monday CRM Zoho CRM الخ..............................

الشركاء الأساسيون
Key Partners

طبيعة الشراكة	معلومات حول الشركاء	الشركاء
Enabling secure electronic payments on the Via platform	A leading company in processing bank transactions and e-payments in Algeria via CIB cards.	SATIM
Ensuring fast and reliable delivery of orders to customers	A leading company in express delivery and logistics services throughout Algeria, specializing in e-commerce	Yalidine

structure Costs هيكل التكاليف

400,000 DZD	تكاليف التعريف بالمنتج أو المؤسسة Frais d'établissement
	تكاليف الحصول على العدادات (الماء- الكهرباء ....) Frais d'ouverture de compteurs (eaux-gaz-....)
Figma Professional ~10,000 – 18,000 DZD GitHub 10,000 – 20,000 DZD Adobe Creative Cloud 80,000 – 120,000 DZD Online Courses (Udemy, Coursera) ~5,000 – 15,000 DZD/course	تكاليف (التكوين- برامج الاعلام الالي المختصة) Logiciels, formations
100000 DZD	Dépôt marque, brevet, modèle تكاليف براءة الاختراع و الحماية الصناعية و التجارية
	Droits d'entrée تكاليف الحصول على تكنولوجيا او ترخيص استعمالها
	Achat fonds de commerce ou parts شراء الأصول التجارية أو الأسهم
	Droit au bail الحق في الإيجار
Commercial Registration 32000 DA	Frais de dossier رسوم إيداع الملفات
	Frais de notaire ou d'avocat تكاليف الموثق-المحامي-.........
	Enseigne et éléments de communication تكاليف التعريف بالعلامة و تكاليف قنوات الاتصال
	Achat immobilier شراء العقارات
	الأعمال والتحسينات الاماكن Travaux et aménagements

structure Costs هيكل التكاليف

Mid-high performance laptops(10): 80,000 – 120,000 DZD each	Matériel الآلات- المركبات- الاجهزة
Desks, chairs, shelves, etc. (for ~10 people) :250,000 – 400,000 DZD	Matériel de bureau تجهيزات المكتب
	Stock de matières et produits تكاليف التخزين
400,000 DZD	Trésorerie de départ التدفق النقدي(الصندوق) الذي تحتاجه في بداية المشروع.

المجموع: **2350000 DZD**

نفقاتك أو التكاليف الثابتة الخاصة بمشروعك

Assurances التأمينات	
Téléphone, internet الهاتف و الانترنت	Internet Pack 1.2 GB 4200 DZD/month
Autres abonnements اشتراكات أخرى	E-payment integration SMS Notification Service (API)
Carburant, transports الوقود و تكاليف النقل	
Frais de déplacement et hébergement تكاليف التنقل و المبيت	
Eau, électricité, gaz فواتير الماء – الكهرباء- الغاز	Electricity from 35000 DA
Mutuelle التعاضدية_الاجتماعية	
Fournitures diverses لوازم متنوعة	
Entretien matériel et vêtements صيانة المعدات والملابس	
Nettoyage des locaux تنظيف المباني	
Budget publicité et communication ميزانية الإعلان والاتصالات	2500000 DZD

المجموع:

مصادر الإيرادات
Revenue Stream

20%	Apport personnel ou familial المساهمة الشخصية أو العائلية
	Apports en nature (en valeur) التبرعات العينية
	Prêt n°1 (nom de la banque) قرض رقم 1 اسم البنك
	Prêt n°2 (nom de la banque) قرض رقم 2 اسم البنك
	Prêt n°3 (nom de la banque) قرض رقم3 اسم البنك
	Subvention n°1 (libellé) منحة 1
	Subvention n°2 (libellé) منحة 2
	Autre financement (libellé) تمويل آخر

المجموع:

رقم الأعمال

بيع المنتج في السنة الأولى

Votre chiffre d'affaires de la première année

بيع المنتج في السنة الأولى	متوسط أيام العمل في الشهر
الشهر 1Mois	20
الشهر 2Mois	20
الشهر 3Mois	20
الشهر 4Mois	20
الشهر 5Mois	20
الشهر 6Mois	20
الشهر 7Mois	20
الشهر 8Mois	20
الشهر 9Mois	20
الشهر 10Mois	20
الشهر 11Mois	20
الشهر 12Mois	20

المجموع:

رقم الأعمال

بيع المنتج في السنة الثانية

Votre chiffre d'affaires de la deuxième année

متوسط أيام العمل في الشهر	بيع المنتج في السنة الثانية
20	الشهر 1Mois
20	الشهر 2Mois
20	الشهر 3Mois
20	الشهر 4Mois
20	الشهر 5Mois
20	الشهر 6Mois
20	الشهر 7Mois
20	الشهر 8Mois
20	الشهر 9Mois
20	الشهر 10Mois
20	الشهر 11Mois
20	الشهر 12Mois

المجموع:

رقم الأعمال

بيع المنتج في السنة الثالثة

Votre chiffre d'affaires de la troisième année

متوسط أيام العمل في الشهر	بيع المنتج في السنة الثالثة
20	الشهر 1Mois
20	الشهر 2Mois
20	الشهر 3Mois
20	الشهر 4Mois
20	الشهر 5Mois
20	الشهر 6Mois
20	الشهر 7Mois
20	الشهر 8Mois
20	الشهر 9Mois
20	الشهر 10Mois
20	الشهر 11Mois
20	الشهر 12Mois

المجموع:

تطور حجم رقم الأعمال في السنة

- النسبة المئوية للزيادة في حجم الأعمال بين السنة 1 والسنة 2؟
- النسبة المئوية للزيادة في حجم الأعمال بين السنة 2 والسنة 3 ؟

حاجتك لرأس المال العامل

30 يوم	متوسط مدة الاعتمادات الممنوحة للعملاء بالأيام Durée moyenne des crédits accordés aux clients en jours
30 يوم	متوسط مدة ديون الموردين بالأيام Durée moyenne des dettes fournisseurs en jours

رواتب الموظفين و مسؤولين الشركة

رواتب الموظفين Salaires employés	Developer : 80000 DA/month UI/UX Designer : 60000 DA/month Marketing Manager : 50000 DA/month Customer Support/Technical Assistant : 40000 DA/month
صافي أجور المسؤولين Rémunération nette dirigeant	

Business Model Canvas

Designed for:	Designed by:	Date:	Version:

Key Partners

Who are our Key Partners? Who are our key suppliers? Which Key Resources are we acquiring from partners? Which Key Activities do partners perform?
MOTIVATIONS FOR PARTNERSHIPS: Optimization and economy, Reduction of risk and uncertainty, Acquisition of particular resources and activities

Key Activities

What Key Activities do our Value Propositions require? Our Distribution Channels? Customer Relationships? Revenue streams?

CATEGORIES:
Production, Problem Solving, Platform/Network

Key Resources

What Key Resources do our Value Propositions require? Our Distribution Channels? Customer Relationships Revenue Streams?

TYPES OF RESOURCES:
Physical, Intellectual (brand patents, copyrights, data), Human, Financial

Value Propositions

What value do we deliver to the customer? Which one of our customer's problems are we helping to solve? What bundles of products and services are we offering to each Customer Segment? Which customer needs are we satisfying?

CHARACTERISTICS: Newness, Performance, Customization, "Getting the Job Done", Design, Brand/Status, Price, Cost Reduction, Risk Reduction, Accessibility, Convenience/Usability

Customer Relationships

What type of relationship does each of our Customer Segments expect us to establish and maintain with them? Which ones have we established? How are they integrated with the rest of our business model? How costly are they?

Channels

Through which Channels do our Customer Segments want to be reached? How are we reaching them now? How are our Channels integrated? Which ones work best? Which ones are most cost-efficient? How are we integrating them with customer routines?

Customer Segments

For whom are we creating value? Who are our most important customers? Is our customer base a Mass Market, Niche Market, Segmented, Diversified, Multi-sided Platform

Cost Structure

What are the most important costs inherent in our business model? Which Key Resources are most expensive? Which Key Activities are most expensive?
IS YOUR BUSINESS MORE: Cost Driven (leanest cost structure, low price value proposition, maximum automation, extensive outsourcing), Value Driven (focused on value creation, premium value proposition).
SAMPLE CHARACTERISTICS: Fixed Costs (salaries, rents, utilities), Variable costs, Economies of scale, Economies of scope

Revenue Streams

For what value are our customers really willing to pay? For what do they currently pay? How are they currently paying? How would they prefer to pay? How much does each Revenue Stream contribute to overall revenues?
TYPES: Asset sale, Usage fee, Subscription Fees, Lending/Renting/Leasing, Licensing, Brokerage fees, Advertising
FIXED PRICING: List Price, Product feature dependent, Customer segment dependent, Volume dependent
DYNAMIC PRICING: Negotiation (bargaining), Yield Management, Real-time-Market

Business Model Canvas

Designed for:	Designed by:	Date:	Version:
Via	Baghdad Amine		

Key Partners

- Local Product Brands & Manufacturers (to promote made-in-Algeria products).
- SATIM – Official Algerian payment gateway provider.
- Cloud Providers – AWS, Azure (for hosting and infrastructure).
- Content Creators & Designer.
- Technology Partners(AI/ML Platform , SMS service provider).
- Delivery & Logistics Companies (Yalidine..etc).
- Marketing Agencie.
- Startup Incubators.

Key Activities

- Platform development (AI systems, APIs, frontend/backend)
- UI/UX design and iteration.
- Payment & Logistics Integration.
- Community building & user support.
- Marketing and campaign management.
- Partner onboarding & collaboration.

Key Resources

- Skilled dev team (DL, backend, frontend).
- High-performance laptops & IT equipment.
- Cloud servers & storage.
- Office space and equipment.
- Software licenses and development tools.
- Financial capital (startup funding or equity).

Value Propositions

- Provides personalized product recommendations using AI and session-based algorithms.
- Built specifically for the Algerian market with multilingual support and local culture in mind.
- Fully responsive and works seamlessly on mobile, tablet, and desktop.
- Helps users and businesses save time with a smooth and efficient experience.
- Features a modern, intuitive, and visually appealing interface.

Customer Relationships

- Support through email, social media, and messaging apps.
- Regular engagement through newsletters, promotions, and updates.
- Feedback collection and continuous improvement based on reviews.
- Community building via social media, forums, and live events.

Channels

- Responsive web platform accessible from mobile and desktop.
- Social media platforms: Facebook, Instagram, TikTok, LinkedIn.
- Optimized presence through search engines and paid ads.

Customer Segments

- Online shoppers in Algeria.
- Small businesses needing visibility.
- Brands seeking smart marketing solutions.

Cost Structure

- Salaries and freelance contracts.
- Equipment (computers, furniture).
- Subscriptions (GitHub, Adobe, Figma , courses ,SMS provider).
- Branding, marketing & domain costs.
- server costs.

Revenue Streams

- Subscription plans for sellers.
- Advertising space for brands and partners.
- Featured listing fees for better product visibility.

TechnoFoster

Contact Us

+213 46 25 61 33

incubator@univ-tiaret.dz

fb.com/techno.foster.incubator