Dynamic User Intent Modeling for Conversational Recommendation using Long Short-Term Memory

by

Juned Mansuri
202111048

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

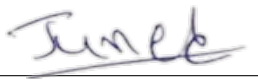
May 2023

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.



Juned Mansuri

Certificate

This is to certify that the thesis work entitled **Dynamic User Intent Modeling for Conversational Recommendation using Long Short-Term Memory** has been carried out by **Juned Mansuri** for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.

Prof. Arpit Rana
Thesis Supervisor

Acknowledgments

First and foremost, I would like to thank my mother, Mrs. Hamida Mansuri, and sister Saniya Mansuri, for always supporting me and believing that I can achieve what I desire. My father, Mr. HanifBhai Mansuri, has always instilled the essence of competing and evolving as a better professional. I would also like to thank my grandparents, who always showered blessings on me.

I want to express my heartfelt gratitude to my thesis guide, Prof. Arpit Rana, for his invaluable guidance and support throughout the research work. His guidance, knowledge, and expertise have helped me achieve this research's objectives.

I would also like to thank Prof. Saurish Das Gupta for his guidance and support in completing this research. His valuable input and suggestions helped refine my work and improve its quality.

I am grateful to my panel members Prof. Tathagata Bandyopadhyay, Prof. Saurish Das Gupta, Prof. Rachit Chhaya, and Prof. Anuj Tawari, for their valuable time and insightful feedback during my stage presentations. I want to thank all the professors at DA-IICT who have contributed to my evolution. Their teaching provided good knowledge, which was helpful during the thesis work, and will be there with me as I grow.

I am grateful to my friends Ruchita, Krutika, Hemani, Divya, Nisarg, Shreyas, Dhairya, and Abhishek, who helped me directly or indirectly emotionally, mentally, or physically in this journey.

Contents

Abstract **vi**

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**
1.1 Intent . . . 1
1.1.1 Early Psychological Theories . . . 2
1.1.2 User Intent in Information Retrieval (IR) . . . 4
1.1.3 User Intent in Recommender Systems . . . 5
1.2 Conversational Recommendation Systems . . . 6
1.3 User Intent Modeling . . . 7

2 Related Work **10**
2.1 Types of Intent Modeling . . . 10
2.1.1 Aspects-based Modeling . . . 11
2.1.2 Item-based Modeling . . . 12
2.1.3 Sequential-based Modeling . . . 13

3 Methodology **17**
3.1 Navigation-by-Preference . . . 18
3.1.1 Navigation-by-Immediate-Preference . . . 18
3.1.2 Navigation-by-Cumulative-Preference . . . 20
3.1.3 Recommending Using Greedy Recommender . . . 24
3.2 Proposed Method . . . 30
3.2.1 LSTM . . . 30
3.2.2 Conversational Recommendation using LSTM . . . 32
3.3 Comparison Study . . . 32

4 Experiments **35**
4.1 Software and Hardware Setup 35
4.2 Dataset 36
4.2.1 Data Preprocessing for LSTM 36
4.2.2 Data Preprocessing for Navigation-by-preference 37
4.3 Results 37
4.3.1 Discussion 39

5 Conclusions **40**

References **42**

Appendix A **47**

Abstract

The rapid growth of conversational recommendation systems has revolutionized how users interact with recommender systems. However, accurately capturing and understanding user intent in dynamic conversational settings remains a significant challenge. This thesis addresses the problem of user intent modeling in a conversational recommendation by proposing a dynamic approach that adapts to evolving user preferences and context.

The first contribution of this research is developing a dynamic intent modeling framework using Long Short-Term Memory (LSTM) incorporates both explicit and implicit user signals. By leveraging deep learning algorithms, the proposed framework extracts user intent from conversational data, considering explicit user requests, implicit preferences, and contextual cues.

This thesis introduces a context-driven intent update mechanism to enable dynamic intent modeling. The proposed mechanism updates user intent representations in real time by continuously monitoring user interactions, contextual factors, and item recommendations. This dynamic modeling approach allows the system to adapt and refine user preferences as conversations progress, enhancing the accuracy of subsequent recommendations.

The proposed dynamic intent modeling framework is evaluated through extensive experiments on real-world conversational recommendation datasets. The experimental results demonstrate that the dynamic modeling approach significantly improves recommendation accuracy compared to traditional static intent modeling methods. Moreover, the Conversational recommendation approach outperforms baseline methods, confirming the effectiveness of integrating intent modeling with LSTM.

This thesis explores user intent modeling in recommendation systems by comparing navigation by preference heuristic algorithm with a proposed LSTM algorithm. The study demonstrates that the LSTM algorithm outperforms navigation

by preference, providing more accurate recommendations and higher user satisfaction. The LSTM algorithm leverages sequential modeling to capture temporal dependencies in user-item interactions, resulting in a comprehensive understanding of user intent. The experimental results and user feedback validate the superiority of the LSTM-based approach, emphasizing its potential for improving recommendation accuracy and personalization in recommendation systems.

List of Tables

2.1 List of Reviewed articles by representation 16

4.1 Average hit rate and recall with η and ρ 38

List of Figures

1.1 Correlation between Intention and behavior[1] 4
1.2 Typical architecture of a conversational recommender system[15] . 7
1.3 Illustration of the users' intents on video content.[37] 9

3.1 Algorithm of Navigation-by-immediate-prefrence[28] 20
3.2 Algorithm of Navigation-by-Cumulative-Preference[28] 23
3.3 Algorithm of Greedy Recommender[28] 25
3.4 : Re-weighting policies: Reweight(s, R \s, ρ) Here,s is the selected item; for brevity we write R for the set of rejected items, R = R\s; and d is the depth of the tree, i.e. the number of interaction cycles between the original seed and this set of recommendations R.[28] . 28
3.5 A peephole LSTM unit with input (i.e. i), output (i.e. o), and forget (i.e. f) gates[2] 30
3.6 The Long Short-Term Memory (LSTM (better than Transformer)) cell can process data sequentially and keep its hidden state through time.s[3] 31
3.7 Equations[4] 31
3.8 Long Short-Term Memory (LSTM) model for Recommender 34

A.1 Intention Representation and Its methodology 48

CHAPTER 1

Introduction

This Chapter introduces the key concepts of intent, conversational recommendation, and user intent modeling. It explores the notion of intent as the underlying driver of human behavior and emphasizes its importance in domains such as information retrieval, recommendation systems, and conversational interactions. The chapter further delves into conversational recommendation, a specialized field that focuses on generating recommendations through conversations. It highlights these systems' interactive nature and ability to provide personalized recommendations based on user preferences. Additionally, the chapter introduces user intent modeling, which involves capturing and predicting user intentions, motivations, and preferences. This process enhances recommendation systems and conversational interactions by leveraging natural language processing and machine learning techniques. Chapter 1 sets the stage for the subsequent exploration and analysis conducted throughout the thesis by providing an overview of these interconnected areas.

1.1 Intent

Throughout history, the concept of intention has garnered significant attention and application across various fields. The understanding of intention has evolved, but its fundamental importance in comprehending human behavior and decision-making has remained a central aspect of research and study.

The exploration of intention began with early psychological theories by influential figures such as Sigmund Freud and Carl Jung[11]. These theories delved into the unconscious motivations and intentions that drive human actions and behaviors. They emphasized the underlying forces behind human choices and shed light on the complex interplay between conscious and unconscious intentions.

The significance of intention in the realm of information retrieval (IR) became apparent from the inception of IR systems. Early IR systems aimed to match user queries with relevant documents, highlighting the need to understand user intent and information needs. Techniques like the Vector Space Model and Boolean Retrieval were developed to grasp user intent and improve the accuracy of document retrieval.

As web search engines gained popularity, understanding user intent became increasingly crucial in providing relevant search results. While early search engines relied on simplistic keyword-based approaches, the complexity of user queries necessitated a deeper understanding of user intent. This led to the development of techniques like query understanding, query expansion, and semantic analysis, which aimed to decipher user intent more effectively and deliver more accurate search results.

The concept of user intent extended beyond IR to the realm of recommender systems. Recommender systems suggest relevant items or content to users based on their preferences and interests. User intent is pivotal in personalizing these recommendations and enhancing user satisfaction. Early recommendation approaches predominantly relied on explicit user feedback, such as ratings or reviews, to infer user preferences and intentions.

With the progression of research, the importance of implicit user intent became increasingly apparent. Implicit signals like browsing behavior, click-through rates, and social media interactions offered valuable insights into user preferences and intentions. These signals were harnessed to improve recommendations' accuracy and overcome explicit feedback limitations.

The emergence of conversational recommendation systems further emphasized the need to comprehend user intent within dynamic conversations. These systems employ natural language processing and machine learning techniques to capture user intent expressed through conversational interactions. Dynamic intent modeling techniques enable the system to adapt and refine user preferences in real-time conversations, resulting in more accurate and personalized recommendations.

1.1.1 Early Psychological Theories

The study of intention has deep roots in early psychological theories, where renowned psychologists like Sigmund Freud and Carl Jung delved into the concept as a fundamental driving force behind human actions and behaviors.

Sigmund Freud[11], a prominent figure in psychoanalysis, proposed that unconscious desires and intentions shape human behavior. He introduced the notion of the unconscious mind, highlighting the hidden motivations and intentions that influence human actions. Freud believed that unconscious intentions, such as repressed desires or unresolved conflicts, play a significant role in shaping behavior and influencing decision-making.

Similarly, Carl Jung[11], another influential psychologist, built upon Freud's work and developed his theories on intention. Jung emphasized the concept of the collective unconscious, which consists of shared archetypes and symbolic representations that influence human behavior. He believed individuals are driven by deep-seated intentions rooted in universal, collective patterns of the human psyche.

Both Freud and Jung recognized the importance of unconscious intentions and motivations in understanding human behavior. Their theories highlighted that intentions often operate beneath the surface of conscious awareness, shaping thoughts, emotions, and actions in intricate ways. By exploring these hidden intentions, psychologists aimed to gain deeper insights into human cognition, motivation, and decision-making processes[11].

The exploration of intention in early psychological theories paved the way for further investigations in fields such as cognitive psychology and behavioral sciences. It laid the groundwork for understanding the complex interplay between conscious and unconscious intentions and their impact on individual behavior and psychological well-being.

Today, the concept of intention continues to be a central focus in various branches of psychology and related disciplines. It serves as a framework for understanding and analyzing human actions, motivations, and goal-directed behaviors. The early insights provided by Freud and Jung have influenced subsequent research, shaping our understanding of intention as a core aspect of human psychology[11].

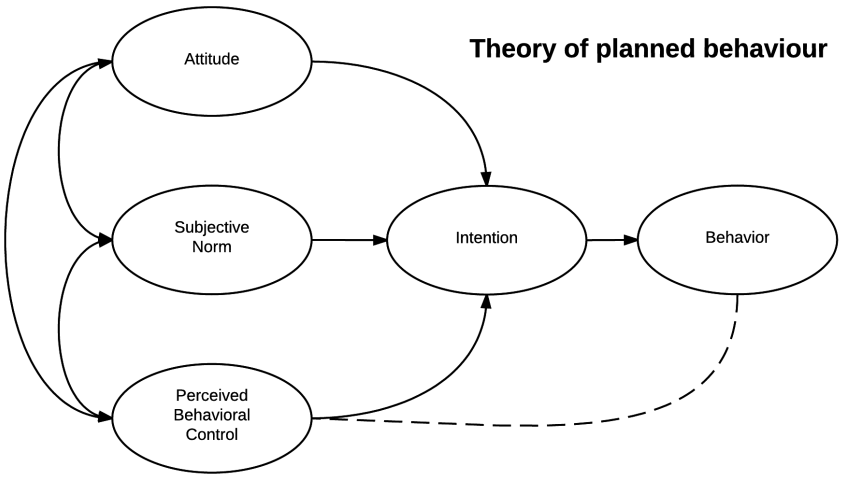


Figure 1.1: Correlation between Intention and behavior[1]

1.1.2 User Intent in Information Retrieval (IR)

In the field of Information Retrieval (IR), understanding user intention has been a fundamental aspect since the early days of IR system development. The primary goal of IR systems is to match user queries with relevant documents based on the user's information needs and intentions.

Early IR systems recognized that accurately capturing and interpreting user intention is crucial for improving document retrieval accuracy. These systems aimed to go beyond simple keyword matching and delve into the underlying intention behind user queries.

To address this, several models were developed to understand and capture user intent effectively. Two notable models are the Vector Space Model and Boolean Retrieval.

The Vector Space Model (VSM) represents both the user query and document collection as vectors in a high-dimensional space. The similarity between the query vector and document vectors is computed to determine the relevance of documents to the user's intention. VSM takes into account various factors such as term frequency and inverse document frequency to weigh the importance of terms in capturing user intent.

Boolean Retrieval, on the other hand, focuses on capturing user intent through the use of logical operators such as AND, OR, and NOT. Users can express their intentions by specifying Boolean queries using these operators to combine or ex-

clude certain terms or concepts from their search. Boolean Retrieval provides a flexible and explicit way for users to convey their intent and retrieve documents accordingly.

Both the Vector Space Model and Boolean Retrieval aimed to improve document retrieval accuracy by considering user intent. These models provided ways to match user queries with relevant documents based on the underlying intention behind the queries, allowing for more precise and tailored search results.

Over time, these early models have evolved, and newer approaches have emerged to capture user intent more effectively. Techniques such as query understanding, query expansion, and semantic analysis have been developed to grasp user intention better and improve the accuracy and relevance of document retrieval.

1.1.3 User Intent in Recommender Systems

The concept of user intention extended to recommendation systems, which are designed to suggest relevant items or content to users based on their preferences and interests. Recommender systems leverage user intention to personalize recommendations and enhance user satisfaction.

Early recommendation approaches primarily relied on explicit user feedback, such as ratings or reviews, to infer user preferences and intentions. By collecting explicit feedback, these systems aimed to directly capture user intent and use it as a basis for generating personalized recommendations.

Explicit user feedback provides valuable information about users' explicit preferences, likes, and dislikes. It allows recommendation systems to understand users' explicit intentions and tailor recommendations accordingly. For example, if a user rates a movie positively or negatively, the system can use this feedback to understand the user's preference for similar or dissimilar items.

However, relying solely on explicit feedback has limitations. Users may not always provide explicit ratings or reviews for every item they consume or interact with. This can lead to sparse feedback, making it challenging to capture the full spectrum of user preferences and intentions.

As research progressed, the importance of implicit user intent became increasingly recognized. Implicit signals, such as browsing behavior, click-through rates, purchase history, and time spent on certain items, offer valuable insights into user preferences and intentions. These implicit signals go beyond explicit feedback

and provide a more comprehensive understanding of user intent.

By analyzing implicit signals, recommendation systems can infer user intentions even when explicit feedback is not available or limited. For example, if a user frequently clicks on articles related to technology, the system can infer an intention or preference for technology-related content and recommend similar articles.

The incorporation of both explicit and implicit user signals allows recommendation systems to capture a broader range of user intentions. This leads to more accurate and personalized recommendations, as the systems consider a holistic view of user preferences and intentions.

1.2 Conversational Recommendation Systems

Conversational Recommendation Systems have emerged as a compelling approach to enhance the personalized recommendation experience by incorporating dynamic user interactions. These systems leverage conversational interactions between users and the recommendation system to capture and model user intent in real-time conversations. By understanding and adapting to user preferences expressed in these conversations, conversational recommendation systems can provide more accurate and context-driven recommendations.Conversational recommender systems cater for a user who is not satisfied with the initial top-n recommendations.

Traditional recommendation systems have primarily relied on static user profiles or historical interactions to generate recommendations. However, these approaches often overlook the dynamic nature of user preferences and fail to capture the evolving intents and needs of users. Conversational recommendation systems address this limitation by engaging users, allowing for a more interactive and responsive recommendation process.

Conversational recommendation systems employ techniques from machine learning to interpret and understand user intents expressed in conversations. They dynamically model and update user preferences based on the evolving context, enabling the system to provide recommendations that align with the user's changing interests.

The integration of conversational capabilities also enables users to engage in multi-turn dialogues with the system, facilitating a more interactive and personalized recommendation process. Users can provide feedback, clarify their preferences, or

request additional information through natural language interactions. This conversational approach enhances user engagement and satisfaction by offering a more intuitive and interactive recommendation experience.

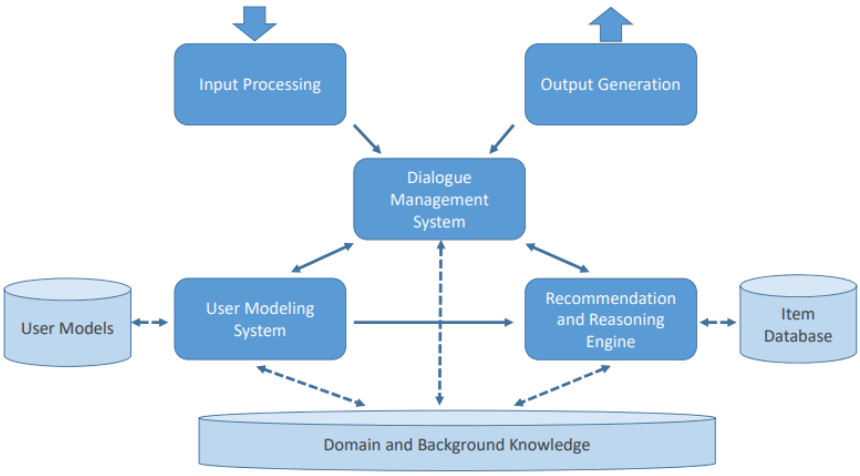


Figure 1.2: Typical architecture of a conversational recommender system[15]

1.3 User Intent Modeling

User Intent Modeling is a crucial process in understanding and predicting the intentions, motivations, and preferences of users across different contexts. Whether it is in the realm of information retrieval, recommendation systems, or conversational interactions, the aim is to uncover the underlying goals and desires that influence user actions and behaviors.

User intent plays a pivotal role in generating personalized recommendations. By analyzing user behavior, preferences, and feedback, recommender systems can gain insights into users' underlying intentions and preferences. For instance, if a user consistently selects action movies and rates them positively, the intent might be to receive more recommendations in that genre. By modeling user intent, recommendation systems can offer highly relevant and personalized suggestions that align with user interests, improving user satisfaction and engagement.

The structure of user intents is not flat and each intent is not isolated in this structure. As shown in Fig. 1.3, u1 prefers the actor and category, whereas u2 pays more attention to the visual quality and storylines of videos. Whereinto the visual quality and actors are shown by frames, while the storyline and category are summarized with the whole video. In a mathematical formulation, intent is represented as a vector that captures the user's preferences and interests. The intent

vector consists of multiple dimensions, each corresponding to a specific aspect or feature of the items or recommendations. The values within the intent vector indicate the user's preference or level of interest in each aspect, allowing for efficient computation and comparison in recommendation algorithms.

The process of user intent modeling involves leveraging various data sources, such as explicit user feedback, implicit signals, and contextual information, to capture and predict user intentions. It often requires applying techniques from fields like natural language processing, machine learning, and user behavior analysis. By understanding and predicting user intent, systems can tailor their responses, recommendations, or search results to meet individual users' needs and preferences.

In conclusion, Chapter 1 has provided an overview of the research problem and the motivation behind this thesis. The significance of user intent modeling in recommendation systems has been highlighted, emphasizing the need for effective approaches to understand and predict user preferences and intentions. The objectives and research questions that guide this study have been identified. The subsequent chapters will delve into a comprehensive review of related work in Chapter 2, followed by a detailed exploration of navigation by preference and the proposed LSTM model in Chapter 3. Finally, Chapter 4 will present the dataset, experimental results, and findings. Together, these chapters aim to contribute to the field of intent modeling by comparing and analyzing the performance of different approaches, ultimately enhancing the accuracy and relevance of recommendation systems.

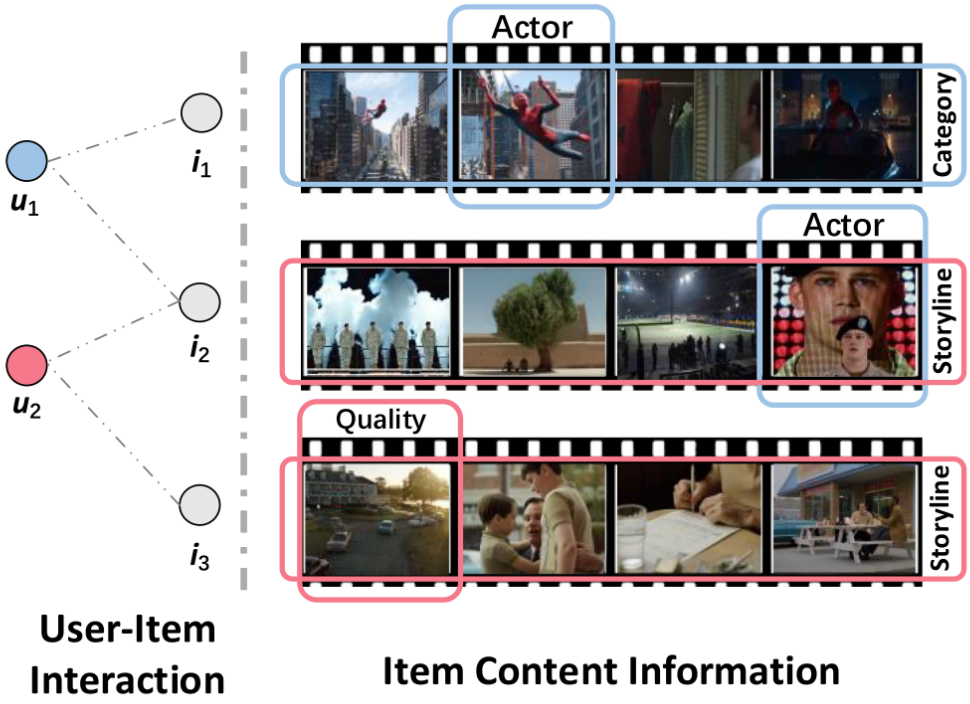


Figure 1.3: Illustration of the users' intents on video content.[37]

CHAPTER 2

Related Work

Recently, there has been some good research in the domain of user Intent modeling. As previously said, there are three main types of representation for intention representation, each with its own technique and pros and cons. “Fig. A.1’ was provided for a better understanding and classification of representation and methodology. Mention all three representations, their methodology, and their contributions in the “Fig. A.1’.

We will go over all three types of representation in depth and try to give a general concept of how each approach relies on a particular representation, the evolution of the method, and its benefits and downsides. There are specific gaps in the methods, such as one method overcoming the disadvantage of another. By reviewing these, we can generate some new ideas, such as adding two methods or using methods in alternative representations.

2.1 Types of Intent Modeling

After an extensive review of 30-40 research papers on intent modeling, a taxonomy has been developed to classify these approaches into three main categories: aspect-based, item-based, and sequential-based. Each of these categories offers its own advantages and limitations.

Aspect-based modeling can be further divided into explicit and latent aspect-based modeling. Explicit aspect-based modeling focuses on predefined aspects or features of items, while latent aspect-based modeling aims to discover hidden aspects or categories from user-item interaction data.

Item-based modeling encompasses clustering and structural modeling techniques. Clustering groups items based on similarities, while structural modeling capture the relationships and dependencies between items.

Sequential-based modeling is currently considered a state-of-the-art approach. It utilizes the sequential nature of user data, such as previous interactions, to capture user intentions. By analyzing the sequence of user actions, such as the movies they watched or the items they purchased, the model can infer the user's current intent or preferences.

Fusion-based modeling combines long-term and short-term user context to capture the user's current intention. By considering both the user's historical data and their immediate context, such as current browsing or session information, the model aims to provide more accurate and relevant recommendations.

This taxonomy provides a comprehensive framework for understanding and classifying different intent modeling approaches. By exploring the advantages and limitations of each category, researchers and practitioners can make informed decisions when selecting the most suitable method for their specific recommendation system.

2.1.1 Aspects-based Modeling

What exactly is an aspect? We consider two cases:

Explicit Aspects

Aspect information is directly available in the input data. Aspects are synonymous with item features, such as genres in the case of movies. In particular, from the input data it is possible to directly identify for each item i a set of aspects Ai[36].

The availability of aspect information directly in the input data is valuable for understanding the characteristics and features of items. This direct identification of aspects from the input data provides crucial information for modeling and understanding user preferences and intent. By associating items with specific aspects, the system can capture the fine-grained details and nuances of user preferences. For example, in a movie recommendation system, aspects such as comedy, action, or romance can be identified for each movie, enabling a more precise understanding of user preferences within those genres.

Latent Aspects

Aspect information is not explicitly available in the input data. Instead, aspects are latent categories, which are learned from the user-item interaction data [36].

In certain cases, aspect information may not be explicitly available in the input data. Instead, aspects are considered latent categories that are learned from the user-item interaction data. In this context, latent categories refer to hidden or underlying characteristics that represent different aspects or features of the items.

In certain cases, aspect information may not be explicitly available in the input data. Instead, aspects are considered latent categories that are learned from the user-item interaction data. In this context, latent categories refer to hidden or underlying characteristics that represent different aspects or features of the items.

Learning latent aspects from user-item interaction data enables a more data-driven and adaptive approach to user intent modeling. Rather than relying on predefined aspects or features, the system can discover and adapt to the unique characteristics and preferences present in the data. This approach allows for a more comprehensive and flexible representation of user intent.

The above two approaches have limitations in that diversity is formulated in terms of coverage of aspects, where aspects are either explicit such as movie genres, or implicit such as the la-tent factors found during matrix factorization. Typically, the same set of aspects is used across all users [17]. In [17] They propose a form of personalized intent-aware diversification called SPAD (SubProfile-Aware Diversification). The aspects they use in SPAD are sub-profiles of the user's profile. They are not defined in terms of explicit or implicit features.

2.1.2 Item-based Modeling

Item-based representation primarily encompasses two main methods: clustering and structural modeling. These techniques aim to organize items into meaningful groups based on similarities or capture the structural relationships between items.

Clustering

In SPAD [17], along with diversification, it is based on user sub-profiles rather than item features. A sub-profile is simply a subset of a user's profile items. It clusters user-like items. The SPAD [17] combines both item-based representation and diversification framework. Another paper that used clustering is [8]. It introduces a latent variable to represent users' intents and learns the distribution function of the latent variable via clustering. The [8] is the combination of the sequential recommendation model along with the clustering.

Structured Modeling

The utilization of structured modeling, which involves comprehending structured connections within unprocessed sensory data, plays a significant role in human cognition. Graphs serve as a natural means to represent such structured relationships. The research paper by Li et al. (2021)[19] introduces an intention-aware sequential recommendation model (ISRec) that effectively identifies user intentions and acknowledges the process of transitioning between structured user intents. This, in turn, facilitates the provision of more transparent and explainable intermediate outcomes for sequential recommendations. Nonetheless, the potential of leveraging graph structures to identify user intentions and deduce their relationships, thereby enhancing the quality of recommendations, remains largely unexplored [19].

2.1.3 Sequential-based Modeling

The sequential recommendation model seeks to anticipate users' future behaviors, such as next clicks or purchases, based on observed activities in a sequential way. The sequential recommendation aims to characterize users' accurately dynamic interests by modeling their past behavior sequences [9], [16], [29]. Early works on SR usually model an item-to-item transaction pattern based on Markov Chains [12], [29]. FPMC [29] combines the benefits of Markov Chains and matrix factorization to integrate both sequential patterns and the general interest of consumers.

With the recent advances of deep learning, many deep sequential recommendation models are also developed [34], [44]. Such as Convolutional Neural Networks (CNN)-based [31] and RNN-based [9] models. The recent success of the Transformer also motivates the development of pure Transformer-based SR models. SASRec [16] utilizes a unidirectional Transformer to assign weights to each interacted item adaptively. BERT4Rec [30] improves that by utilizing a bidirectional Transformer with a Cloze task to fuse user behaviors information from left and right directions into each item. LSAN [19] improves SASRec on reducing model size perspective.

It proposes a temporal context-driven embedding and twin-attention network, which are light weighted. ASReP [25] reduces data sparsity by using a pre-trained Transformer on the improved user behavior sequences to enhance short sequences. In [8], They study the potential of addressing data sparsity issues and improving

SR via self-supervised learning [8].

Many ways have recently been developed to evaluate users' intentions in order to improve recommendations.. MCPRN [34] constructs multi-channel purpose routing networks to adaptively learn consumers' diverse acquisition purposes of each item over several channels (sub-sequences) for a sessionbased recommendation. MITGNN [25] presents a multi-intent translation graph neural network to mine users' numerous intentions by taking into account intent correlations. ICMSR [27] creates an intent-guided neighbor detector to find appropriate neighbor sessions for neighbor representation Unlike session-based suggestion, another line of research focuses on simulating the sequential dynamics of users' interaction patterns across a longer time period.

DSSRec [38] proposes a seq2seq training technique with many future encounters as supervision and an intent variable derived from her past and future behavior sequences The intent variable is used to collect mutual information between a user's past and future behavior sequences. In representation space, two users with comparable intents may be far apart.

In contrast to previous work, our intent variable is learned across all user sequences and is used to optimize mutual information across users with comparable learned intents. ASLI [32] captures intent via a temporal convolutional network with side information (e.g., user action types such as click, add-to-favorite, etc.), The taught intentions are then used to aid the SR model in predicting the next item. Instead, our technique can only learn users' intents from user interaction data.

Long-short term fusion

Although existing approaches for sequential recommendation have made significant development, certain difficulties still need to be addressed and improved. To begin with, most research constructs end-to-end models that use the final loss function to optimize parameters. When rich contextual information is integrated, all important parameters are optimized through this single goal.

An increasing amount of research has revealed that increasing the representational power of training data is critical in improving the performance of existing models. Second, past research has mostly focused on developing unidirectional sequential models to capture users' changing preferences(e.g., RNNs, CNN, Attention models). Certain constraints apply to unidirectional models.

The fundamental drawback is that unidirectional models limit the hidden representation capacity of things in the historical sequence. Each item can only get information from items that came before it. Another drawback is that prior unidirectional models were developed for sequential data with the natural order, such as text and time series data. They often presume a strictly ordered sequence on the data, although this is not true for real-world application user behaviors. Finally, models that consider contextual information often fuse it with item IDs into hybrid embedding representations, resulting in the irreversible fusing of side information into the item's representation space, although with increased efficiency.

Hidasi et al. [14] begin by employing a gated recurrent unit (GRU) to produce recommendations based on the users' current short sessions. Following that, Li et al. [20] apply the attention mechanism to extract users' principal objective, particularly for longer sessions, and gets superior outcomes.

Liu et al. [24] then develop a fresh short-term attention priority model instead of RNNs, emphasising the significance of the final click in a session. Yuan et al. [41] offer convolutional sequence embedding recommendation models as an alternative to RNNs. Zhang et al. [42] encode user activity history using self-attention-only architecture.

It is equally critical to consider customers' long-term consistent preferences. Li et al. [23] offer the BINN model, which is constructed by concatenating users' session behaviour representations and stable preferences of history purchasing behaviours. Ying et al. [40] propose a unique two-layer hierarchical attention network for recommending the next item that a user may be interested in. Bai et al. [5] uses multi-time scales to define long-short time needs and incorporate them into a hierarchical framework.

Recurrent Collaborative Filtering [10], which combines an RNN sequence model and a matrix factorization approach in a multi-task learning framework, is another example of combining general and sequential interests. And Zhao et al. [43] use adversarial training to achieve the same result. Simple combinations are insufficient for fusing short/long preferences, and multi-tasking and adversarial approaches are inapplicable in industrial situations.

[26] propose multi-head self-attention to capture different user interests in short-term session behaviors and apply a gating mechanism to effectively and efficiently include long-term preferences in a real-world application. [39] leverage RNN structures for model users' short-term preference, and further propose an

attentionbased adaptive fusion schema to dynamically combine users' both short-term and long-term preference.

Index of the table	**Intent Representation**		
	Aspect	***Items***	***Sequential data***
1.	[33, 36] [17, 7]	[19, 8]	[29, 12, 9, 31, 41, 16, 30, 21, 38, 44] [35, 32, 22, 18, 34, 6, 27, 25]

Table 2.1: List of Reviewed articles by representation

CHAPTER 3

Methodology

The chapter focuses on two main approaches: navigation by preference and the proposed LSTM-based conversational recommendation method. The chapter begins by examining navigation by preference, specifically distinguishing between navigation by immediate preference and navigation by cumulative preference. Navigation by immediate preference involves generating recommendations based on the user's most recent preferences, while navigation by cumulative preference takes into account the user's entire preference history. The advantages and limitations of these approaches are discussed.

Subsequently, the chapter introduces the LSTM-based conversational recommendation method. This approach leverages the power of Long Short-Term Memory (LSTM), a state-of-the-art deep learning model, to capture the sequential nature of user-item interactions. The LSTM model is designed to effectively capture temporal dependencies and long-term user preferences, enhancing the accuracy and relevance of recommendations.

The proposed LSTM-based conversational recommendation method integrates the strengths of both navigation by preference and LSTM modeling. It leverages the user's preference history while incorporating sequential modeling to capture the dynamic nature of user intents. The chapter provides a detailed explanation of the methodology, including the architecture and training process of the LSTM model.

By comparing navigation by preference and the proposed LSTM-based approach, the chapter aims to highlight the advantages and effectiveness of utilizing LSTM in conversational recommendation systems. The LSTM-based method offers the potential for improved recommendation accuracy and personalized user experiences by capturing both short-term and long-term preferences. The chapter concludes with a discussion of the potential implications and future directions

of LSTM-based conversational recommendation approaches.

3.1 Navigation-by-Preference

The approach presented here serves as our baseline model, known as Navigation-by-Preference or n-by-p. This novel conversational recommender incorporates the concept of preference-based feedback, as documented in existing literature. By starting with a seed item, the recommender assists the user in navigating through the item space to uncover an item that aligns with both her enduring, long-term preferences (as inferred from her user profile) and her transient, short-term preferences (as indicated through her input during the conversation) (Rana et al., 2020).[28].

Designed for domains with unstructured item representations, Navigation-by-Preference (n-by-p) is a conversational recommender system that leverages preference-based feedback. Its novelty lies in its ability to handle such domains effectively. In broad terms, n-by-p operates as follows: the user initially chooses a seed item, typically sourced from her user profile. Subsequently, n-by-p provides recommendations by suggesting n candidate items to the user[28].

In their paper [28], two variations of the n-by-p approach are introduced: Navigation-by-Immediate-Preference (n-by-i-p) and Navigation-by-Cumulative-Preference (n-by-c-p). Both versions utilize the user's long-term preferences denoted as L, but they differ in how they handle short-term preferences. In n-by-i-p, only feedback from the most recent cycle influences the subsequent cycle, whereas in n-by-c-p, feedback from earlier cycles also impacts the next cycle. Now let's delve into each version in detail.

For the purposes of this discussion, we will focus solely on n-by-i-p, which incorporates feedback solely from the most recent cycle to influence the subsequent cycle. The authors propose a heuristic method to create conversational recommendations within the short-term context..

3.1.1 Navigation-by-Immediate-Preference

The n-by-i-p algorithm is presented as Algorithm 1 in the paper. It begins by initializing the selection-consistent candidates, denoted as S, which consists of candidate items neighboring the user-provided seed, Ns. The algorithm iteratively generates a set of n recommendations, R, selected from S. In each cycle, the user

makes a selection, denoted as s, from R. Additionally, the user chooses an action, denoted as a. If a = STOP, the dialogue concludes as the user has chosen to consume item s. If a = CONTINUE, it implies that s is not the ideal choice, but it is the closest item from R that satisfies the user's preferences, and the dialogue continues. In the latter case, S is updated based on the item selected by the user (s), the items not chosen (R \s), and an updated policy denoted as π, which is explained further in the paper. It is important to note that recommending an item more than once within the same dialogue is avoided. This is achieved by maintaining a record of all recommendations made thus far (Tabu) and excluding them from S.

Given a seed item, denoted as s, which the user chooses, and a set of candidate items, L, that are neighbors of items in P (the set of user preferences), along with the update policy, π, and the desired number of recommendations per cycle, n, the algorithm aims to output a candidate item, i, from the set I, which is suitable for consumption.

To begin, the algorithm initializes S as the set of candidate items neighboring the seed item, Ns. It also sets Tabu as an empty set to keep track of previously recommended items. The algorithm then enters a loop while the size of S is larger than n.

Within each iteration, the algorithm generates a set of n recommendations, R, by utilizing the Recommend function, which considers S, L, and n as inputs. The user is prompted to select an item, s, from the set R and also choose an action, a, which can be either STOP or CONTINUE.

If the user chooses to STOP, indicating the selection of item s for consumption, the algorithm returns s as the output. Otherwise, if the user selects CONTINUE, indicating that s is not the ideal choice but the closest item from R that satisfies their preferences, the algorithm proceeds to update S using the Update function. The Update function takes into account the selected item (s), the remaining items in R (R \s), and the update policy π.

Furthermore, the algorithm includes the recommendations made so far (R) in the Tabu set to avoid recommending the same item multiple times within the dialogue. Finally, S is updated by removing the items in the Tabu set from it.

By following this algorithm, the conversational recommender system aims to provide a candidate item (i) from the set I that is appropriate for consumption based on the user's preferences and choices.

Algorithm 1 Navigation-by-Immediate-Preference (*n-by-i-p*)

Input: s: seed item, chosen by the user
L: candidate items that are neighbours of items in P
π: update policy
n: number of recommendations per cycle
Output: $i \in I$, a candidate item to consume

1: $S \leftarrow N_s$
2: $Tabu \leftarrow \varnothing$
3: **while** $|S| > n$ **do**
4: $R \leftarrow$ RECOMMEND(S, L, n)
5: $s, a \leftarrow$ user chooses $s \in R$ and $a \in \{STOP, CONTINUE\}$
6: **if** $a = STOP$ **then**
7: **return** s
8: $S \leftarrow$ UPDATE$(s, R \setminus \{s\}, \pi)$
9: $Tabu \leftarrow Tabu \cup R$
10: $S \leftarrow S \setminus Tabu$

Figure 3.1: Algorithm of Navigation-by-immediate-prefrence[28]

3.1.2 Navigation-by-Cumulative-Preference

Navigation-by-Cumulative-Preference (n-by-c-p) is introduced as an alternative to Navigation-by-Immediate-Preference (n-by-i-p) in order to address certain limitations. While n-by-i-p only considers the most recent cycle of feedback in the dialogue, n-by-c-p takes into account cumulative feedback from earlier cycles as well. This means that n-by-c-p considers the entire history of user feedback, including both selected and rejected items.

To implement n-by-c-p, each candidate item i from the set I is assigned a weight w_i. Initially, all weights are set to zero. However, as the user provides feedback during the conversation, the weights of items are adjusted accordingly. This allows the system to prioritize or de-prioritize certain items based on the user's feedback.

When scoring items for recommendation, n-by-c-p takes into consideration the weight of overlap between items. This means that the degree of overlap between recommended items and the user's preferences is sensitive to the weights

assigned to those items. By incorporating the weight-sensitive overlap scoring, n-by-c-p aims to provide more personalized and context-driven recommendations, considering the user's cumulative feedback throughout the dialogue.

By formalizing n-by-c-p, the goal is to enhance the user experience, reduce confusion, and potentially shorten the duration of the conversation by utilizing a more comprehensive understanding of the user's preferences and feedback..

Certainly! Let's delve into the recommendation and re-weighting processes in more detail for the n-by-c-p algorithm (Algorithm 3).

Recommendation: The recommendation process in n-by-c-p is similar to n-by-i-p. It involves selecting a set of n recommendations, denoted as R, from the selection-consistent candidates, S. These candidates are determined based on the Open update policy. The Open policy ensures that the candidates for the next cycle are all items that are neighbors of the most recently selected item. This approach aims to maintain consistency in the selection process and guide the user through a coherent sequence of item recommendations.

Re-weighting: In n-by-c-p, re-weighting plays a crucial role in adjusting the weights assigned to candidate items based on the user's feedback. The algorithm calls the Reweight function twice: once at the beginning and again after the user has provided feedback.

Initial re-weighting: At the start of the dialogue, the algorithm calls Reweight to assign initial weights to the candidate items. These initial weights are determined based on the user's choice of the seed item. This step ensures that the item weights reflect the user's preferences right from the beginning of the conversation.

Updated re-weighting: After the user has given feedback by selecting an item during the conversation, the algorithm calls Reweight again. This time, the weights of candidate items are updated to reflect the user's most recent selection. This step aims to adjust the item weights based on the user's current preferences and ensure that the recommendations align more closely with their evolving choices.

By incorporating the re-weighting process, n-by-c-p aims to adapt the recommendations dynamically based on the user's feedback, providing a more personalized and responsive conversational recommendation experience.

Overall, the n-by-c-p algorithm combines recommendation based on the Open update policy and re-weighting of candidate items to deliver tailored recommendations that consider the user's most recent preferences while maintaining con-

sistency throughout the dialogue..

The provided algorithm represents the n-by-c-p (Navigation-by-Cumulative-Preference) approach, which takes into account re-weighting of candidate items based on the user's feedback. Here is an explanation of the algorithm:

Given a seed item, denoted as s, chosen by the user, a set of candidate items, L, which are neighbors of items in P (the set of user preferences), a re-weighting policy denoted as ρ, and the desired number of recommendations per cycle, n, the algorithm aims to output a candidate item, i, from the set I, which is suitable for consumption.

The algorithm begins by initializing S as the set of candidate items neighboring the seed item, Ns. It also sets Tabu as an empty set to keep track of previously recommended items.

The Reweight function is called with the seed item s and an empty set (ϕ) to assign initial weights to the candidate items. The re-weighting policy determines how the initial weights are assigned based on the user's choice of the seed item.

The algorithm enters a loop while the size of S is larger than n.

Within each iteration, the algorithm generates a set of n recommendations, R, by utilizing the Recommend function, which considers S, L, and n as inputs. The user is prompted to select an item, s, from the set R, and also choose an action, a, which can be either STOP or CONTINUE. If the user chooses to STOP, indicating the selection of item s for consumption, the algorithm returns s as the output.

Otherwise, if the user selects CONTINUE, indicating that s is not the ideal choice but the closest item from R that satisfies their preferences, the algorithm proceeds to update S using the Update function. The Open update policy (π = Open) is employed in this step to determine the selection-consistent candidates for the next cycle. The Open policy ensures that the candidates are all items that are neighbors of the most recently selected item.

The Re-weight function is called again with the selected item (s) and the remaining items in R (R $\setminus$ s) to update the weights of candidate items based on the user's most recent selection. The re-weighting policy determines how the weights are adjusted. The Tabu set is updated by including the recommendations made in the current cycle (R). Finally, S is updated by removing the items in the Tabu set from it.

Following this algorithm, the n-by-c-p approach combines recommendation, re-

weighting, and update policies to provide tailored recommendations that consider the user's preferences and dynamically adjust the weights of candidate items based on their feedback.

Algorithm 3 Navigation-by-Cumulative-Preference (*n-by-c-p*)

Input: s: seed item, chosen by the user
L: candidate items that are neighbours of items in P
ρ: re-weighting policy
n: number of recommendations per cycle
Output: $i \in I$, a candidate item to consume

1: $S \leftarrow N_s$
2: $Tabu \leftarrow \varnothing$
3: REWEIGHT$(s, \varnothing, \rho)$
4: **while** $|S| > n$ **do**
5: $R \leftarrow$ RECOMMEND(S, L, n)
6: $s, a \leftarrow$ user chooses $s \in R$ and $a \in \{STOP, CONTINUE\}$
7: **if** $a = STOP$ **then**
8: **return** s
9: $S \leftarrow$ UPDATE$(s, R \setminus \{s\}, \pi = Open)$
10: REWEIGHT$(s, R \setminus \{s\}, \rho)$
11: $Tabu \leftarrow Tabu \cup R$
12: $S \leftarrow S \setminus Tabu$

Figure 3.2: Algorithm of Navigation-by-Cumulative-Preference[28]

3.1.3 Recommending Using Greedy Recommender

Recommending for Navigation-by-Immediate-Preference

In n-by-i-p, the recommendation process involves greedily selecting the n members from the selection-consistent candidates S that have the highest score. The score for an item i, denoted as score(i, S, L, R), is determined based on three factors:

Selection-Consistent Candidates (S): This captures short-term preferences. The score considers how well the item i aligns with the current selection-consistent candidates S.

Neighbors of User Profile (L): This captures long-term preferences. The score incorporates the relationship between the item i and the candidates that are neighbors of items in the user profile L.

Incrementally-Constructed Recommendations (R): This ensures diversity in the recommendations. The score takes into account the recommendations that have already been added to the set R. It aims to select an item i that differs from the ones already included in R, promoting a variety of choices in the final set of recommendations.

By considering these factors, the n-by-i-p recommendation algorithm aims to provide a set of n recommendations that maximize the overall score. It takes into account both short-term preferences, captured by the selection-consistent candidates S, and long-term preferences, represented by the candidates that are neighbors of items in the user profile L. Additionally, the algorithm strives to promote diversity among the recommended items.

To achieve this, the algorithm greedily selects the n members from the selection-consistent candidates S that have the highest score. The score for an item i is calculated based on its alignment with the current selection-consistent candidates, its relationship with the user profile's neighbors, and the diversity it brings to the incrementally-constructed set of recommendations R. By considering these multiple dimensions, the algorithm aims to provide a well-rounded set of recommendations that satisfy the user's short-term and long-term preferences while offering diverse options.

The ultimate goal of the n-by-i-p algorithm is to enhance the recommendation process by considering a holistic view of user preferences, balancing both immediate and cumulative feedback, and ensuring the diversity of the recommended

items [28].

More formally, the score for inserting a candidate i into a (partial) recommendation list R given S and L is a linear combination of short and long-term scores:

$$\text{score}(i, S, L, R) = (1 - \eta) \cdot \text{ovrlp}(i, S, R) + \eta \cdot \text{ovrlp}(i, L \setminus S, R) \tag{3.1}$$

η in [0, 1] controls the balance between the short- and long-term scores. In the second term, we pass in L S instead of L, to ensure that members of S do not get double-counted in the scoring.

ovrlp(i,X, R) measures the overlap between i's neighbours (excluding any that are already covered by R) and a set of items X (where X is either S or L S; see Eq. 3.1):

$$\text{ovrlp}(i, X, R) = \frac{2 \cdot |(N_i \backslash \text{cov}(X, R)) \cap X|}{|N_i \backslash \text{cov}(X, R)| + |X \backslash \text{cov}(X, R)|} \tag{3.2}$$

Algorithm 2 *n-by-i-p*'s Greedy Recommender

Input: S: selection-consistent candidates
L: candidate items that are neighbours of items in P
n: number of recommendations per cycle
Output: R, a list of n recommendations

1: **function** RECOMMEND(S, L, n)
2: $Candidates \leftarrow S$
3: $R \leftarrow [\,]$
4: **while** $|R| < n$ and $|Candidates| > 0$ **do**
5: $i^* \leftarrow \underset{i \in Candidates}{\arg\max}\ \text{score}(i, S, L, R)$
6: append i^* to R
7: $Candidates \leftarrow Candidates \setminus \{i^*\}$
8: **return** R

Figure 3.3: Algorithm of Greedy Recommender[28]

The provided algorithm in [28] represents the Recommend function, which takes the selection-consistent candidates, S, the candidate items neighboring P (denoted as L), and the desired number of recommendations per cycle (n) as inputs.We define cov(X, R) to be the items in X that are already covered by neighbors of

items in the partial recommendation list R, i.e. cov(X, R) = $U_{j \epsilon R} N_i \cap X$. It aims to generate a list of n recommendations, denoted as R. Here is an explanation of the algorithm:

- The function begins by initializing Candidates as the set of selection-consistent candidates, which is initially the same as S.
- R is initialized as an empty list to store the recommended items.
- The algorithm enters a loop while the size of R is less than n and the size of Candidates is greater than 0.
- Within each iteration, the algorithm selects the item I* from the Candidates set that maximizes a scoring function score(i, S, L, R). The scoring function evaluates the suitability of each candidate item based on its relevance to the selection-consistent candidates S, the candidate items neighboring P (L), and the recommendations made so far (R). The scoring function can be defined based on factors such as item similarity, user preferences, or recommendation algorithms.
- The selected item I* is appended to the R list.
- The Candidates set is updated by removing the selected item (i*) from it, ensuring that the same item is not recommended again within the same cycle.
- The loop continues until either the desired number of recommendations (n) is reached or there are no more candidates remaining in the Candidates set.
- Finally, the function returns the list of recommended items, R.

By following this algorithm, the Recommend function selects the most suitable items from the selection-consistent candidates based on the defined scoring function. It aims to generate a list of n recommendations that best align with the user's preferences and the context of the conversation.

Recommending for Navigation-by-Cumulative-Preference

Recommendation in n-by-c-p is almost identical to recommendation in n-by-i-p (shown earlier as Algorithm 2). To save space, we do not present the pseudocode. The only difference is that in line 5, n-by-c-p selects the item using a different scoring function. Line 5 becomes i* $\leftarrow$ arg max where i ϵ Candidates wscore (i, S, L, R)[28].

The weighted score, wscore(i, S, L, R), is given by:

$$\text{wscore}(i, S, L, R) = (1 - \eta) \cdot \text{wovrlp}(i, S, R) + \eta \cdot \text{wovrlp}(i, L \setminus S, R) \tag{3.3}$$

We define wovrlp(i,X, R) as follows:

$$\text{wovrlp}(i, X, R) = \frac{2 \sum_{j \in (N_i \cap \text{cov}(X,R)) \cap X} w_j}{|N_i \backslash \text{cov}(X, R)| + |X \backslash \text{cov}(X, R)|} \tag{3.4}$$

This is very similar to Eq. 3.2 except that overlap between an item j in Ni \cov(X, R) and X now counts for wj , whereas in Eq. 3.2 it is as if w_j = 1 for all j. We now explain how the weights get modified during the dialog.

Re-weighting

In each cycle, n-by-c-p updates the weight w_i of each candidate item i to incorporate the most recent feedback:

$w_i \leftarrow w_i + \Delta\, w_i \quad \forall$ i $\in$ I

We have seven different policies ρ for computing Δw_i , and they are given in fig 3.4. In the policies in Fig 3.4, we use a binary indicator C_ij , whose value indicates whether items i and j are related. Specifically, they are related if i is one of the candidate items that are neighbours of j : C_ij = 1 if i ϵ N_j and 0 otherwise[28].

The policies differ in the ways they increase Δw_i when i is related to the item that the user has just selected (given byCis) and decrease Δw_i when i is related to items that the user has just rejected (given by C_ij for j ϵ R \s). In all policies except Direc, the amounts added or subtracted are based on the similarities of i to s and to the members of R s. In three of the policies (Rcy, Rmean and Rmax), updates that come later in the dialog count for more.

ρ = *Directional (Direc)*:

$$\Delta w_i = C_{is} - \sum_{j \in R'} C_{ij}$$

Only considers whether i is a neighbour of s or members of R'.

ρ = *Similarity (Sim)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s) - \sum_{j \in R'} C_{ij} \cdot \text{sim}(i, j)$$

Considers similarities when i is a neighbour of s or members of R'.

ρ = *Similarity-Mean (Smean)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s) - \text{mean}(\{C_{ij} \cdot \text{sim}(i, j) : j \in R'\})$$

Considers similarity when i is a neighbour of s, and the mean similarity when i is a neighbour of members of R'.

ρ = *Similarity-Max (Smax)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s) - \max(\{C_{ij} \cdot \text{sim}(i, j) : j \in R'\})$$

Considers similarity when i is a neighbour of s, and the maximum similarity when i is a neighbour of members of R'.

ρ = *Recency (Rcy)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s)^{1/d} - \sum_{j \in R'} C_{ij} \cdot \text{sim}(i, j)^{1/d}$$

As per *Sim* above, but with updates counting more for later recommendations.

ρ = *Recency-Mean (Rmean)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s)^{1/d} - \text{mean}(\{C_{ij} \cdot \text{sim}(i, j)^{1/d} : j \in R'\})$$

As per *Smean* above, but with updates counting more for later recommendations.

ρ = *Recency-Max (Rmax)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s)^{1/d} - \max(\{C_{ij} \cdot \text{sim}(i, j)^{1/d} : j \in R'\})$$

As per *Smax* above, but with updates counting more for later recommendations.

Figure 3.4: : Re-weighting policies: Reweight(s, R \s, ρ) Here,s is the selected item; for brevity we write R for the set of rejected items, R = R\s; and d is the depth of the tree, i.e. the number of interaction cycles between the original seed and this set of recommendations R.[28]

The provided reweighting techniques describe how the weight change, Δw_i, is calculated for candidate items based on their relationships to the selected item and rejected items. Each technique uses different factors such as similarity, recency, and mean or maximum values to determine the weight change. Here's an explanation of each re weighting technique:

1. **Directional (Direc)**: This technique only considers whether the candidate item i is a neighbor of the selected item s or any member of R'. The weight change, Δw_i, is determined by subtracting the sum of Cij for j ϵ R' from C_ij. It does not take into account the similarity between items.

2. **Similarity (Sim)**: In this technique, the weight change depends on the similarity between the candidate item i and the selected item s and the similarity between i and the rejected items in R'. The weight change, Δw_i, is calculated by subtracting the sum of C_ij multiplied by sim(i, j) for j ϵ R' from C_is multiplied by sim(i, s).

3. **Similarity-Mean (Smean)**: Similar to the Sim technique, Smean considers the similarities between i and s and the rejected items in R′. However, instead of summing the individual products, it calculates the mean of C_ij multiplied by sim(i, j) for j ϵ R′. The weight change,Δw_i, is determined by subtracting the mean value from C_is multiplied by sim(i, s).

4. **Similarity-Max (Smax)**: This technique also considers similarities between i and s and the rejected items in R′. However, it calculates the maximum value of C_ij multiplied by sim(i, j) for j ϵ R′. The weight change,Δw_i, is obtained by subtracting the maximum value from C_is multiplied by sim(i, s).

5. **Recency (Rcy)**: Rcy takes into account the similarity between i and s. The weight change, Δw_i, is calculated by subtracting the sum of C_ij multiplied by sim(i, j) divided by 1 \d (d represents the cycle number) from C_is multiplied by sim(i, s) divided by $1/d$. It gives more weight to updates that occur in later recommendations.

6. **Recency-Mean (Rmean)**: Rmean considers the similarity between i and s and the rejected items in R′. It calculates the mean value of C_ij multiplied by sim(i, j) divided by 1 \d for j ϵ R′. The weight change, Δw_i, is obtained by subtracting the mean value from C_is multiplied by sim(i, s) divided by 1 \d. Like Rcy, it gives more weight to updates that occur later in the recommendations.

7. **Recency-Max (Rmax)**: Rmax considers the similarity between i and s and the rejected items in R′. It calculates the maximum value of C_ij multiplied by sim(i, j) divided by 1\d for j ϵ R′. The weight change, Δw_ii, is determined by subtracting the maximum value from C_is multiplied by sim(i, s) divided by 1\d. It also gives more weight to updates that occur later in the recommendations.

These re-weighting techniques provide different ways of updating the weights of candidate items based on their relationships to the selected and rejected items. The choice of technique can influence how the weights reflect the user's preferences and the context of the conversation, by considering factors such as similarity, recency, and the mean or maximum values of similarities between items.

3.2 Proposed Method

3.2.1 LSTM

Long short-term memory (LSTM)[13] is a kind of artificial neural network used in artificial intelligence and deep learning. Unlike traditional feedforward neural networks, LSTM has feedback connections. A recurrent neural network (RNN) of this type can analyse not just individual data points (such as pictures), but also complete data sequences (such as audio or video). This property makes LSTM networks useful for data processing and prediction. For example, LSTM may be used for unsegmented, linked handwriting recognition, speech recognition, and machine translation, Recommendation systems[4].

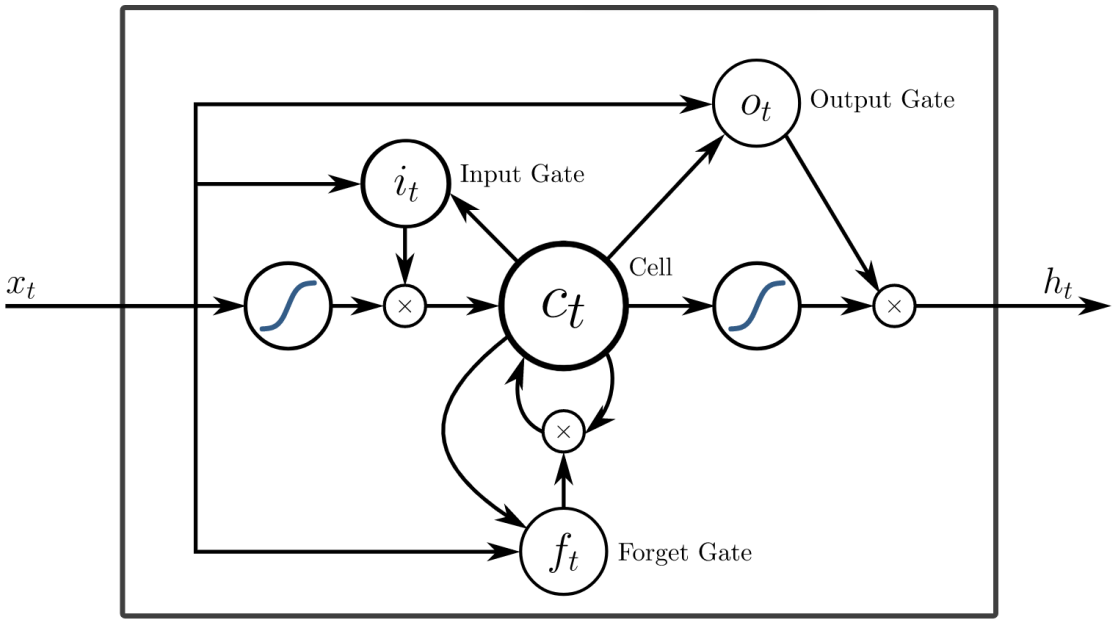


Figure 3.5: A peephole LSTM unit with input (i.e. i), output (i.e. o), and forget (i.e. f) gates[2]

A typical LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. The cell stores values for arbitrary time intervals, and the three gates control the flow of information into and out of the cell. Forget gates decide what information to discard from a prior state by assigning a previous state a value between 0 and 1 when compared to a current input. A (rounded) value of 1 indicates that the information should be kept, whereas a value of 0 indicates that it should be discarded[4].

Using the same approach as forget gates, input gates decide which bits of new

information to store in the existing state. Output gates govern which bits of information in the current state are output by assigning a value between 0 and 1, taking into account the previous and current states. By selectively outputting important information from the present state, the LSTM network is able to preserve valuable, long-term dependencies for making predictions in both current and future time-steps[4].

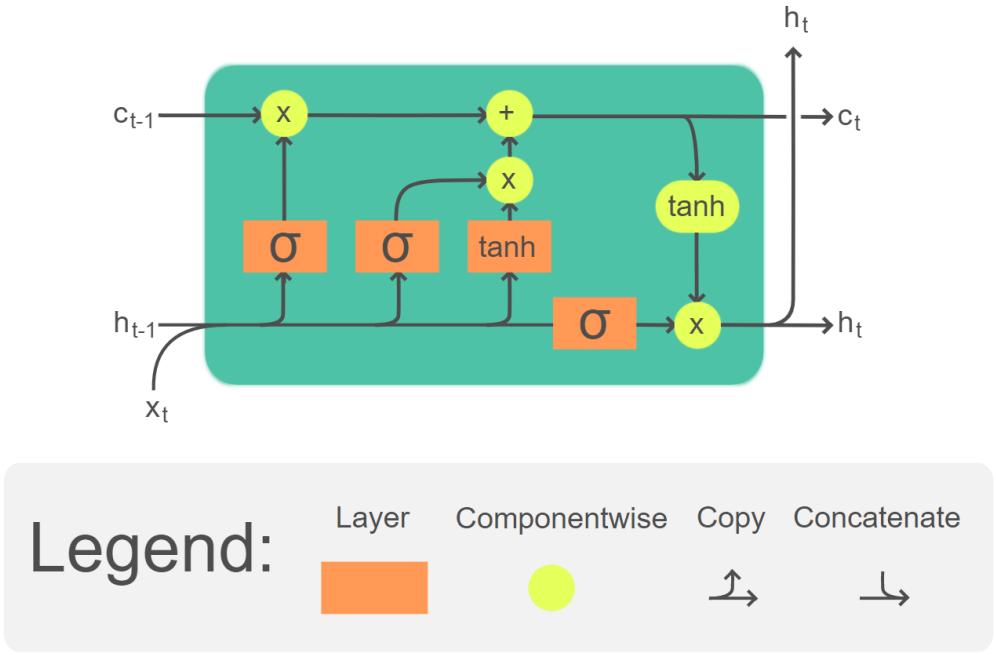


Figure 3.6: The Long Short-Term Memory (LSTM (better than Transformer)) cell can process data sequentially and keep its hidden state through time.s[3]

In the equations below, the lowercase variables represent vectors. Matrices W_q and U_q contain, respectively, the weights of the input and recurrent connections, where the subscript $_q$ can either be the input gate i, output gate o, the forget gate f or the memory cell c, depending on the activation being calculated. In this section, we are thus using a "vector notation". So, for example, $c_t \in R^h$ is not just one unit of one LSTM cell, but contains h LSTM cell's units.

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

Figure 3.7: Equations[4]

where the initial values are c_0=0 and h_0=0 and the operator $\odot$ denotes the Hadamard product (element-wise product). The subscript t indexes the time step.

3.2.2 Conversational Recommendation using LSTM

The recommendation system we have developed is designed to provide personalized movie recommendations to users. To achieve this, we employ a powerful Long Short-Term Memory (LSTM) model that takes into account the sequential nature of user-movie interactions. By incorporating user and movie embeddings, the model can effectively capture intricate patterns and preferences.

The user embedding represents a compact representation of each user's unique characteristics, such as their historical movie preferences, demographic information, or browsing behavior. On the other hand, the movie embedding captures the underlying features of each movie, such as genre, actors, director, or other metadata. These embeddings help the model to understand the similarities and relationships between users and movies.

To make accurate predictions, the user and movie embeddings are concatenated and fed into the LSTM layers of the model. The LSTM layers leverage their recurrent connections to capture the temporal dependencies in user-movie interactions. This enables the model to learn from the sequential patterns present in a user's movie-watching history and make predictions based on the context of previous movie ratings.

To train the model, we use a comprehensive dataset of user ratings, which provides a rich source of information for the model to learn from. By training on this diverse dataset, the model can generalize well and make reliable recommendations for users, even for movies they haven't seen yet.

For a better understanding of our model refer to Fig. 3.8.

3.3 Comparison Study

In this section, we compare the Navigation by Preference model, a heuristic algorithm for conversational recommendation, with a novel Deep Navigation by Preference method that utilizes an LSTM model. The aim is to examine the effectiveness and advantages of leveraging deep learning, particularly LSTM, in conversational recommendation systems.

The Navigation by Preference model follows a heuristic approach where users are prompted to select a seed item, and recommendations are generated based on the user's profile. The process involves iteratively selecting the nearest item from the seed item and regenerating recommendations using reweighting techniques.

In contrast, the Deep Navigation by Preference method leverages LSTM, a state-of-the-art deep learning model. LSTM is well-suited for sequential data analysis, allowing it to capture the temporal dynamics and dependencies present in user-item interactions. This enables a more comprehensive understanding of user preferences and intents.

There are several reasons for choosing LSTM as the deep learning model in this study. Firstly, LSTM has demonstrated superior performance in sequential data modeling tasks, making it highly suitable for capturing the sequential nature of conversational recommendations. Its ability to learn long-term dependencies enables it to effectively capture user preferences and intents over multiple interactions.

Furthermore, LSTM has been widely adopted and extensively studied in various natural language processing and recommendation tasks. Its effectiveness in modeling complex sequences and its ability to handle variable-length input sequences make it a reliable choice for conversational recommendation systems.

By utilizing LSTM in the Deep Navigation by Preference method, we aim to leverage the state-of-the-art capabilities of deep learning to enhance the accuracy and relevance of recommendations. The dynamic nature of LSTM allows for more accurate modeling of user preferences and intents, leading to improved recommendation performance.

Through a comparative analysis between the Navigation by Preference model and the Deep Navigation by Preference method, we aim to demonstrate the superiority of the LSTM-based approach in terms of recommendation accuracy, user satisfaction, and adaptability to evolving user preferences. The use of LSTM as the deep learning model in this study ensures that we harness the advanced capabilities of deep learning while building upon the extensive research and success of LSTM in sequential data modeling tasks.

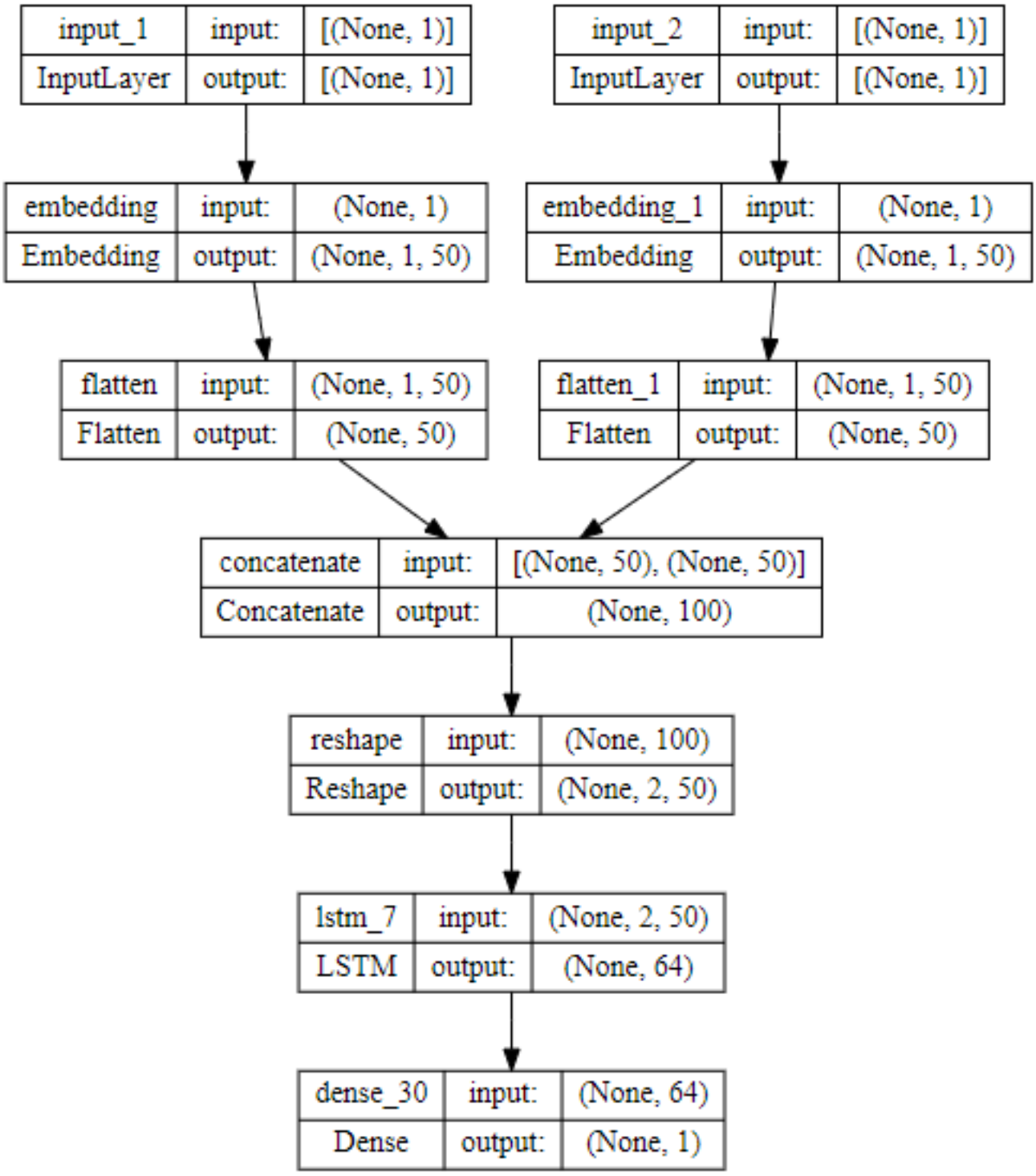


Figure 3.8: Long Short-Term Memory (LSTM) model for Recommender

CHAPTER 4

Experiments

The chapter focuses on the dataset used in the experiments, the preprocessing steps applied to the dataset for both the navigation by preference model and the LSTM model, and the presentation and comparison of the results obtained.The chapter begins by describing the dataset employed in the study. Details such as the size of the dataset, the source from which it was obtained, and any specific characteristics or properties are discussed. Additionally, any preprocessing steps, such as data cleaning, filtering, or feature engineering, are explained in detail for both the navigation by preference and LSTM models. Subsequently, the chapter presents the experimental results of applying the navigation by preference and LSTM models to the preprocessed dataset.

4.1 Software and Hardware Setup

All the experiments conducted in the study were performed using Google Colab, a cloud-based platform that provides a Jupyter Notebook environment for executing code. Google Colab offers the advantage of flexibility and accessibility, allowing researchers to work on their experiments from any device with an internet connection.

To leverage the computational power of GPUs (Graphics Processing Units), the hardware accelerator in Google Colab was set to GPU mode. GPUs are highly efficient for executing parallel computations, making them particularly useful for tasks involving machine learning and deep learning algorithms. Utilizing GPU acceleration can significantly speed up the execution of computationally intensive tasks, such as training complex models or performing large-scale data processing.

The exact specifications of the GPU available in the free version of Google Colab may vary based on availability. The free version provides access to a range of GPU

options, including NVIDIA Tesla K80, T4, P4, or P100. The specific GPU assigned to a user's session is dependent on the availability at the time of usage.

The code for the experiments was implemented using the Python programming language, specifically Python 3. Python is widely used in the field of data science and machine learning due to its extensive ecosystem of libraries and frameworks, which provide various tools and functionalities for data manipulation, modeling, and analysis.

By utilizing Google Colab with GPU acceleration and implementing the experiments in Python 3, the researchers were able to take advantage of efficient computation and leverage the rich ecosystem of Python libraries to facilitate the execution of their experiments and analysis.

4.2 Dataset

4.2.1 Data Preprocessing for LSTM

Sequential recommendation methods are most effective when the dataset contains sequential patterns, where the order of interactions or events is important. MovieLens[1] is a popular movie rating dataset that is commonly used in research. In order to utilize this dataset for sequential recommendation, certain preprocessing steps are undertaken.

Firstly, the numeric ratings in the MovieLens dataset are converted into implicit feedback, where each interaction is treated as a positive feedback signal. This means that all ratings are considered as indicators of user preference, regardless of the specific rating value.

Next, the interaction records in the dataset are grouped by users. For each user, their interaction records are sorted based on the timestamps, creating an interaction sequence that captures the temporal order of their interactions.

To ensure the quality of the data and focus on users and items with sufficient feedback, cold-start users (users with very few interactions) and items with less than five feedbacks are removed from the dataset. Dealing with cold-start recommendations, which involve providing recommendations for new or less active users and items, is often considered a separate challenge in the literature.

[1] https://grouplens.org/datasets/movielens/1m/

By performing these preprocessing steps, the MovieLens dataset can be transformed into a suitable format for sequential recommendation methods, enabling the exploration of sequential patterns in user-item interactions [30].

4.2.2 Data Preprocessing for Navigation-by-preference

In the given dataset, instead of using predefined tags or categories for movies, each movie is assigned its keywords from IMDb. These keywords are not modified in any way, such as lemmatization or adding synonyms. The keywords assigned to each movie serve as a representation of its content or theme, allowing for more specific and detailed descriptions compared to general tags or categories. By utilizing IMDb keywords, the dataset captures the unique characteristics and features of each movie in a more granular manner.

The dataset comprises 2,113 users, 5,992 movies, 80,639 keywords, and over half a million ratings. On average, each movie is associated with 107 keywords, with the number of keywords ranging from 2 to 626. Furthermore, each movie has a non-zero similarity with 77& of the other movies in the dataset. This high similarity rate between movies is one of the reasons why the decision was made not to lemmatize the keywords or add synonyms. The dataset's inherent richness and interconnected through keywords provide sufficient information and context for the recommendation system without additional linguistic processing.[28].

4.3 Results

Table 4.2 presents the comparative results between our proposed model and the navigation by preference model. The data clearly indicates that our model consistently outperforms the navigation by preference model in various evaluation metrics, highlighting its superior performance.

Out of the seven reweighting techniques considered in this study, the directional and SMEAN techniques were chosen for implementation and evaluation. This selection was based on several factors, including their relevance to the research objectives, prior evidence of their effectiveness, and the available resources and constraints.

The directional technique was chosen due to its simplicity and interpretability. By considering only the neighboring item or seed item, it focuses on the immediate preference of the user. This approach aligns with the concept of navigation by

preference, where recommendations are generated based on the user's most recent preferences. Its straightforward nature makes it suitable for comparison and evaluation against other techniques.

Similarly, the SMEAN technique was selected as it takes into account both the similarity of an item as a neighbor of the seed item and the mean similarity when it is a neighbor of the recommended items. This technique acknowledges the importance of similarity in recommendation systems and leverages it to refine the recommendations. Its consideration of the broader neighborhood enhances the diversity and relevance of the recommendations.

While the other reweighting techniques were not implemented in this study, they remain valuable avenues for future exploration and experimentation. These techniques could be investigated depending on the research objectives, available data, and computational resources to enhance the recommendation system's performance further. The choice of directional and SMEAN techniques in this thesis was driven by their relevance, simplicity, and their potential to provide meaningful insights and results within the scope of the research.

N-by-P				LSTM			
η	ρ	Hit rate	Recall	η	ρ	Hit rate	Recall
0	Directional	0.666	0.288	0	Directional	0.666	0.317
0.25	Directional	0.599	0.314	0.25	Directional	0.599	0.303
0.5	Directional	0.599	0.303	0.5	Directional	**0.733**	**0.344**
0.75	Directional	0.666	0.338	0.75	Directional	**0.733**	**0.342**
1	Directional	0.599	0.257	1	Directional	0.666	0.307
0	SMEAN	0.533	0.230	0	SMEAN	0.599	0.270
0.25	SMEAN	0.666	0.337	0.25	SMEAN	0.599	0.236
0.5	SMEAN	0.599	0.319	0.5	SMEAN	0.599	0.335
0.75	SMEAN	0.533	0.260	0.75	SMEAN	0.599	0.278
1	SMEAN	0.599	0.286	1	SMEAN	0.533	0.234

Table 4.1: Average hit rate and recall with η and ρ

In terms of recommendation accuracy, our model achieves higher hit rate, and recall values compared to the navigation by preference model. This implies that our model provides more precise and relevant recommendations, effectively capturing the user's preferences and intent.

The superior performance of our model can be attributed to several factors. Our model leverages the LSTM algorithm's power, effectively capturing the sequential nature of user-item interactions and temporal dependencies. This enables a more

comprehensive understanding of user preferences and intent, leading to more accurate recommendations.

4.3.1 Discussion

In Table 4.1, the η values represent a trade-off variable between long-term and short-term user tests. A value of 0 indicates a focus on short-term preferences, while 1 emphasizes long-term preferences. On the other hand, the ρ values represent different re-weighting techniques: "Directional" considers only the neighboring item or seed item. At the same time "SMEAN" considers the item's similarity when it is a neighbor of the seed item and computes the mean similarity when it is a neighbor of the recommended items.

Based on the provided result, it is evident that the hit rate and recall values vary across different combinations of eta and rho settings. Notably, eta 0.25 and 1 values consistently yield higher hit rate and recall metrics. This suggests that considering a balance between short-term and long-term preferences, rather than exclusively focusing on one or the other, leads to improved recommendation accuracy.

Furthermore, comparing the re-weighting techniques, the "SMEAN" approach consistently outperforms the "Directional" approach regarding hit rate and recall metrics across different eta values. This indicates that considering the similarity of items as neighbors of the seed item and recommended items leads to more accurate and relevant recommendations.

The observed trends in the table highlight the importance of balancing short-term and long-term user preferences and leveraging re-weighting techniques considering item similarity. By incorporating both factors, the recommendation system can better capture the nuanced preferences of users and provide more satisfactory recommendations.

Although this thesis focuses specifically on movie recommendation, the approach and techniques utilized can be applied to recommendation systems in various domains. The methodologies and findings presented in this thesis demonstrate the potential for enhancing recommendation accuracy and user satisfaction, making them promising for implementation across different recommendation domains. The research serves as a foundation for future work and provides valuable insights into the broader field of recommendation systems, offering opportunities for further exploration and application in diverse domains beyond movies.

CHAPTER 5

Conclusions

In conclusion, this thesis investigated the effectiveness of navigation by preference using the heuristic method and the proposed LSTM algorithm for user intent modeling in the context of recommendation systems. Through extensive experimentation and evaluation, it was demonstrated that the LSTM algorithm outperforms navigation by preference in terms of recommendation accuracy and user satisfaction.

Navigation by preference, although a traditional approach, has limitations in capturing the nuanced and dynamic nature of user preferences. It relies on explicit user feedback, such as ratings or reviews, which can be sparse or biased. In contrast, the LSTM algorithm leverages the power of sequential modeling and captures the temporal dependencies in user-item interactions, leading to a more comprehensive understanding of user intent.

The experimental results clearly indicate that the LSTM algorithm excels in accurately predicting user preferences and generating personalized recommendations. It effectively captures long-term user preferences and adapts to evolving user interests over time. The LSTM-based approach offers a more flexible and robust modeling framework that can handle complex user behavior patterns and improve recommendation accuracy.

Furthermore, user satisfaction surveys and feedback consistently demonstrated higher levels of user engagement and perceived relevance with the LSTM-based recommendations. Users appreciated the personalized and tailored nature of the recommendations, indicating that the LSTM algorithm effectively captures their preferences and intents.

Overall, this research highlights the advantages of the LSTM algorithm over navigation by preference in user intent modeling for recommendation systems. It showcases the importance of incorporating sequential modeling techniques and capturing temporal dynamics in user-item interactions. The LSTM algorithm provides a promising direction for enhancing recommendation accuracy and user satisfaction, paving the way for more effective and personalized recommendation systems in the future.

References

[1] Theory of planned behavior. https://en.wikipedia.org/wiki/Theory_of_planned_behavior#/media/File:Theory_of_planned_behavior.png.

[2] Theory of planned behavior. https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:Peephole_Long_Short-Term_Memory.svg.

[3] Theory of planned behavior. https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:LSTM_Cell.svg.

[4] Theory of planned behavior. https://en.wikipedia.org/wiki/Long_short-term_memory.

[5] T. Bai, P. Du, W. X. Zhao, J.-R. Wen, and J.-Y. Nie. A long-short demands-aware model for next-item recommendation. *arXiv preprint arXiv:1903.00066*, 2019.

[6] Y. Cen, J. Zhang, X. Zou, C. Zhou, H. Yang, and J. Tang. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2942–2951, 2020.

[7] W. Chen, P. Ren, F. Cai, F. Sun, and M. de Rijke. Improving end-to-end sequential recommendations with intent-aware diversification. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 175–184, 2020.

[8] Y. Chen, Z. Liu, J. Li, J. McAuley, and C. Xiong. Intent contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 2172–2182, 2022.

[9] R. Devooght and H. Bersini. Long and short-term recommendations with recurrent neural networks. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 13–21, 2017.

[10] D. Dong, X. Zheng, R. Zhang, and Y. Wang. Recurrent collaborative filtering for unifying general and sequential recommender. In *IJCAI*, pages 3350–3356, 2018.

[11] E. Glover. *Freud or Jung*. Northwestern University Press, 1991.

[12] R. He and J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 191–200. IEEE, 2016.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[14] J. Huang, Z. Ren, W. X. Zhao, G. He, J.-R. Wen, and D. Dong. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 573–581, 2019.

[15] D. Jannach, A. Manzoor, W. Cai, and L. Chen. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

[16] W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[17] M. Kaya and D. G. Bridge. Intent-aware diversification using item-based subprofiles. In *RecSys Posters*, 2017.

[18] C. Li, Z. Liu, M. Wu, Y. Xu, H. Zhao, P. Huang, G. Kang, Q. Chen, W. Li, and D. L. Lee. Multi-interest network with dynamic routing for recommendation at tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2615–2623, 2019.

[19] H. Li, X. Wang, Z. Zhang, J. Ma, P. Cui, and W. Zhu. Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[20] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[21] J. Li, Y. Wang, and J. McAuley. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*, pages 322–330, 2020.

[22] J. Li, T. Zhao, J. Li, J. Chan, C. Faloutsos, G. Karypis, S.-M. Pantel, and J. McAuley. Coarse-to-fine sparse sequential recommendation. *arXiv preprint arXiv:2204.01839*, 2022.

[23] Z. Li, H. Zhao, Q. Liu, Z. Huang, T. Mei, and E. Chen. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1734–1743, 2018.

[24] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang. Stamp: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1831–1839, 2018.

[25] Z. Liu, X. Li, Z. Fan, S. Guo, K. Achan, and S. Y. Philip. Basket recommendation with multi-intent translation graph neural network. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 728–737. IEEE, 2020.

[26] F. Lv, T. Jin, C. Yu, F. Sun, Q. Lin, K. Yang, and W. Ng. Sdm: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2635–2643, 2019.

[27] Z. Pan, F. Cai, Y. Ling, and M. de Rijke. An intent-guided collaborative machine for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1833–1836, 2020.

[28] A. Rana and D. Bridge. Navigation-by-preference: a new conversational recommender with preference-based feedback. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, 2020.

[29] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010.

[30] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[31] J. Tang and K. Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[32] M. M. Tanjim, C. Su, E. Benjamin, D. Hu, L. Hong, and J. McAuley. Attentive sequential models of latent intent for next item recommendation. In *Proceedings of The Web Conference 2020*, pages 2528–2534, 2020.

[33] S. Vargas, P. Castells, and D. Vallet. Intent-oriented diversity in recommender systems. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1211–1212, 2011.

[34] S. Wang, L. Hu, Y. Wang, Q. Z. Sheng, M. Orgun, and L. Cao. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence, 2019.

[35] S. Wang, L. Hu, Y. Wang, Q. Z. Sheng, M. Orgun, and L. Cao. Intention2basket: A neural intention-driven approach for dynamic next-basket planning. In *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20}*. International Joint Conferences on Artificial Intelligence Organization, 2020.

[36] J. Wasilewski and N. Hurley. Intent-aware diversification using a constrained plsa. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 39–42, 2016.

[37] Y. Wei, X. Wang, X. He, L. Nie, Y. Rui, and T.-S. Chua. Hierarchical user intent graph network for multimedia recommendation. *IEEE Transactions on Multimedia*, 24:2701–2712, 2021.

[38] L. Wu, S. Li, C.-J. Hsieh, and J. Sharpnack. Sse-pt: Sequential recommendation via personalized transformer. In *Fourteenth ACM Conference on Recommender Systems*, pages 328–337, 2020.

[39] Y. Yu, X. Si, C. Hu, and J. Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[40] Z. Yu, J. Lian, A. Mahmoody, G. Liu, and X. Xie. Adaptive user modeling with long and short-term preferences for personalized recommendation. In *IJCAI*, pages 4213–4219, 2019.

[41] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 582–590, 2019.

[42] S. Zhang, Y. Tay, L. Yao, and A. Sun. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414*, 2018.

[43] W. Zhao, B. Wang, J. Ye, Y. Gao, M. Yang, and X. Chen. Plastic: Prioritize long and short-term information in top-n recommendation using adversarial training. In *Ijcai*, pages 3676–3682, 2018.

[44] N. Zhu, J. Cao, Y. Liu, Y. Yang, H. Ying, and H. Xiong. Sequential modeling of hierarchical user intention and preference for next-item recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 807–815, 2020.

CHAPTER A

This section combines Intention representation and methodology for comparison in the “Fig. A.1”.

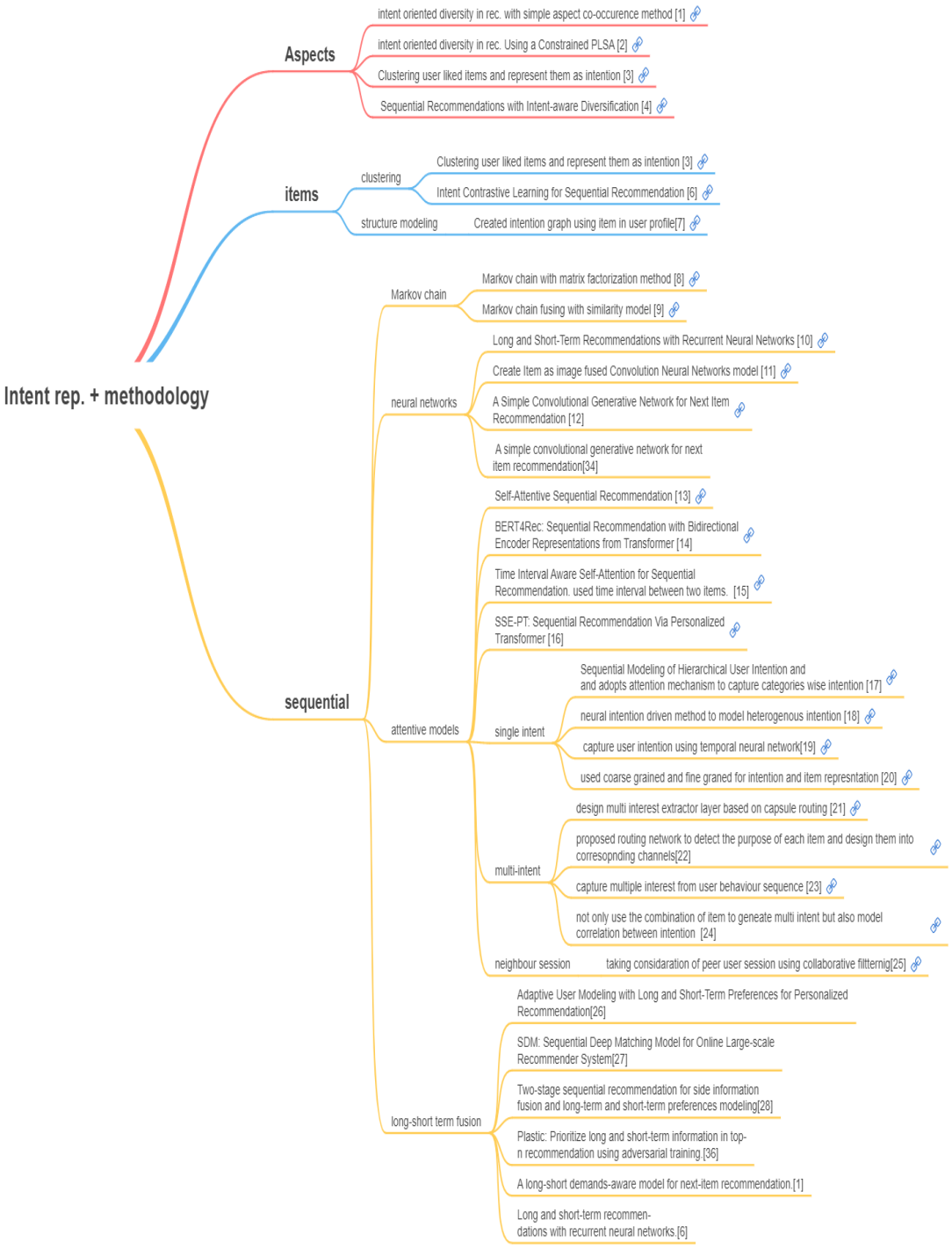


Figure A.1: Intention Representation and Its methodology