Context-Aware and Adaptive Multi-Scale Interest Modeling for Sequential Recommendation

by

Xiaowen Wang

A thesis submitted to the University of Ottawa
in partial fulfillment of the thesis requirement for the degree of

Master of Computer Science

School of Electrical and Computer Science
Faculty of Engineering
University of Ottawa

Examining Committee

The following served on the Examining Committee for this thesis.

External Examiner:	Alan Tsang Assistant Professor School of Computer Science Carleton University
Internal Member(s):	Paria Shirani Assistant Professor School of Electrical Engineering and Computer Science University of Ottawa
Supervisor(s):	Thomas Tran Professor School of Electrical Engineering and Computer Science University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those concerning consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Sequential recommendation aims to predict a user's next interaction by modeling ordered user–item behavior sequences and plays a critical role in modern recommender systems. In real-world scenarios, user behavior is influenced by multiple contextual factors, among which temporal dynamics and item popularity are particularly important. Time intervals between interactions reflect the evolution and decay of user interests, while item popularity introduces frequency bias that may cause recommender systems to overemphasize popular items and underrepresent long-tail preferences. Moreover, user interests naturally exist at different temporal scales: long-term behaviors capture stable and persistent preferences, while recent interactions often reflect short-term, context-dependent intents. Effectively modeling these signals and integrating long-term and short-term interests remains a central challenge in sequential recommendation.

To address these challenges, this thesis presents two sequential recommendation models with increasing modeling capability. The first model, Dual-Gated Time Frequency Co-Modeling for Sequential Recommendation (TiIfSRec), is designed to explicitly incorporate temporal intervals and item-frequency information into long-term and short-term interest modeling. TiIfSRec employs a dual-gated recurrent architecture in which time-interval signals control the decay of historical preferences, while item-frequency signals regulate the direction of state updates to mitigate popularity bias and preserve long-tail information. An attention mechanism is further introduced to highlight informative historical interactions and improve long-range dependency modeling. Experiments on multiple Amazon benchmark datasets demonstrate that TiIfSRec consistently outperforms representative time-aware and popularity-aware baselines, validating the effectiveness of jointly modeling temporal and frequency signals for sequential recommendation.

While TiIfSRec improves long-term and short-term interest modeling, its fusion strategy relies on deterministic mechanisms, which limits flexibility when balancing stable preferences and rapidly changing intents. Motivated by this limitation, the second model, Diffusion-based Long–Short Interest Fusion for Sequential Recommendation (DiffLSRec), formulates long–short interest integration as a generative process. DiffLSRec introduces a diffusion-based framework in which the long-term interest representation serves as a generative prior, and the short-term interest representation is incorporated as conditional guidance during a multi-step denoising process. This progressive fusion strategy enables the model to adaptively adjust the contribution of long-term and short-term interests at different stages, rather than relying on a single static fusion operation. To further enhance contextual modeling and stability, DiffLSRec incorporates token-level contextual enhancement to capture fine-grained recent behavioral patterns, as well as a monotonic

signal-to-noise ratio–adaptive guidance mechanism to regulate the influence of short-term signals throughout the diffusion trajectory. Extensive experiments show that DiffLSRec consistently outperforms strong sequential recommendation baselines, including diffusion-based models, across multiple evaluation metrics.

Overall, this thesis demonstrates that explicitly modeling contextual behavioral signals and progressively integrating user interests across different temporal scales can substantially improve the accuracy, robustness, and adaptability of sequential recommender systems. The proposed models provide complementary contributions, with TiIfSRec offering effective context-aware interest encoding and DiffLSRec introducing a flexible generative paradigm for long–short interest fusion.

Acknowledgements

I am deeply thankful to my supervisor, Professor Thomas Tran, for his guidance, patience, and consistent support throughout my graduate studies. His careful feedback and thoughtful suggestions were invaluable to the development and completion of this thesis. His support helped maintain clarity and direction throughout the research and writing process.

I would also like to thank my family and friends for their understanding and encouragement throughout this journey. Their support, both practical and emotional, provided steady motivation during periods of intensive research and writing. Their patience and encouragement made it possible for me to remain focused and committed during challenging stages of this work.

This thesis reflects the collective support of these individuals, without whom its completion would not have been possible.

I acknowledge the use of AI-assisted tools for writing and editorial support, including grammar correction, sentence refinement, and improving the clarity of technical explanations. All content has been carefully reviewed and validated by the author. The research methodology, experimental design, data analysis, and conclusions presented in this thesis are solely my own responsibility.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Recommender systems have become an essential part of modern e-commerce platforms, enabling users to efficiently explore large-scale and diverse item collections. By modeling users' historical interaction data, Recommender systems aim to capture users' latent preferences and produce personalized recommendation results that are consistent with individual interests. These systems play an important role in supporting user decision-making and improving interaction efficiency. With the continuous expansion of online platforms, the number of available items and the diversity of content have increased substantially, leading to a much larger candidate space for each user. In such settings, users are often required to choose from a large set of alternatives, and identifying items that are relevant to their preferences becomes increasingly challenging without the support of effective algorithmic recommender models.

In real-world applications, user interaction sequences are generated over time and form ordered behavior sequences rather than isolated events. These sequences reflect the evolution of user preferences as new items, contexts, and usage needs change continuously. Therefore, user interests are dynamic and may change at different stages of interaction, making sequence modeling a central problem in recommendation research. Sequential recommendation aims to capture temporal dependencies in the interaction history to improve the next prediction. Compared with static recommendation settings, sequential scenarios require models to consider not only which items users interact with, but also the order of interactions, the temporal intervals, and the evolving patterns that drive changes in user preferences over time.

Traditional recommender systems are generally categorized into content-based methods [25] and collaborative filtering (CF) approaches [44]. Among them, CF-based methods, including matrix factorization and its extensions [1, 42], have been widely adopted due to their effectiveness in modeling user–item relationships from interaction data. These methods learn latent representations of users and items to capture global preference patterns. However, they typically model user interactions as unordered and static, without explicitly considering the temporal structure of user behavior. Such models have limited capacity to reflect how user interests change over time, especially in scenarios where recent behaviors are more informative than earlier interactions.

Different from general recommender systems that treat user preferences as static, sequential recommendation methods focus on modeling ordered user behaviors and the temporal evolution of user preferences over time, leading to increasing research attention [37]. Early sequential approaches are mainly based on recurrent neural networks [3], which update user representations step by step along interaction sequences. These models are effective in capturing short-term dependencies and local temporal patterns in user behavior. For long interaction histories, recurrent architectures often face challenges in modeling long-range dependencies due to gradient vanishing, while their sequential processing nature limits computational efficiency [24]. These limitations motivate the exploration of alternative architectures for sequential recommendation.

More recent studies have adopted self-attention mechanisms to model user interaction sequences. Attention-based models allow each interaction to directly consider other interactions within the sequence, enabling more flexible modeling of long-range dependencies with improved computational efficiency [14]. These attention-based models have been widely adopted and evaluated in sequential recommendation tasks. However, most existing attention-based methods represent user behavior with a single embedding, without explicitly distinguishing stable long-term preferences from transient short-term intents. This unified representation may be insufficient in scenarios where user interests shift rapidly or where recommendation decisions need to balance long-term behavioral patterns with immediate contextual signals.

Recent studies further suggest that user interests exist at different temporal scales. Long term preferences reflect relatively stable behavioral patterns accumulated over extended interaction histories, while short-term intents capture transient and context-dependent needs driven by recent actions. Prior work has shown that modeling both long-term and short-term user preferences can improve sequential recommendation performance, particularly in dynamic settings where user behavior evolves over time [2]. However, most existing approaches rely on static fusion strategies, such as fixed weighting or simple gating mechanisms, which limit their flexibility in balancing long-term and short-term interests under

varying contexts. This observation motivates the exploration of more adaptive mechanisms for integrating user interests at different temporal ranges in sequential recommendation.

Beyond temporal modeling, real-world recommendation data typically exhibits a skewed item distribution, where a small number of popular items account for most user interactions while many items receive relatively few interactions. Such popularity imbalance is widely observed in large-scale recommendation datasets and has been shown to bias recommendation models toward head items, limiting their ability to capture diverse user preferences. Prior studies on long-tail recommendation indicate that adequately modeling infrequent items is crucial for improving recommendation diversity and effectiveness [21]. To mitigate popularity bias, recent work has explored incorporating additional contextual signals, such as item frequency and temporal information, together with more flexible modeling strategies [22]. These studies suggest that jointly accounting for temporal dynamics and popularity-related factors leads to more expressive sequential recommendation models.

Overall, existing studies indicate that effective sequential recommendation relies on a clear understanding of how user preferences evolve over time. User behavior is shaped by multiple interacting factors, including temporal dynamics, item popularity, and the presence of both relatively stable long-term preferences and transient short-term intents. These characteristics suggest that modeling user interests as static or single-scale representations is often insufficient for complex real-world scenarios. Instead, sequential recommendation models should explicitly account for contextual signals and preference dynamics across different temporal scales to more accurately reflect the evolving nature of user interests.

1.2 Motivations

Despite substantial progress in sequential recommendation, effectively modeling evolving user interests in real-world settings remains challenging. As discussed in the overview sections, user behavior is shaped by multiple interacting factors, including temporal dynamics, item popularity, and the interaction between relatively stable long-term preferences and transient short-term intents. These factors jointly influence how users interact with items over time and introduce considerable complexity into the modeling process. However, many existing approaches adopt restrictive modeling assumptions that simplify one or more of these aspects, which limits their ability to capture such complexity in practical recommendation scenarios [27, 37]. There is a growing need to explore more expressive and adaptive sequential recommendation frameworks that can better accommodate diverse behavioral patterns and dynamic preference evolution.

Many existing sequential recommendation models primarily focuses on the order of user interactions, implicitly assuming that interactions occur at uniform or comparable time intervals [17]. In real-world scenarios, however, the temporal gaps between consecutive interactions can vary considerably and often encode important information about user intent and interest intensity. Interactions that occur within short time spans may indicate strong short-term engagement or focused user intent, while longer gaps may reflect shifts in attention, changes in contextual conditions, or evolving user needs. Such temporal irregularity is common in practical recommendation environments, where user activity patterns are often sparse and irregularly distributed over time. Ignoring these temporal signals can lead to incomplete or distorted representations of user behavior, particularly in settings characterized by irregular interaction frequencies or limited observation data, limiting the model's ability to accurately capture dynamic preference evolution.

In addition, real-world recommendation data commonly exhibits popularity imbalance, where a small subset of items accounts for a large proportion of user interactions [39]. This skewed distribution often causes sequential recommendation models to be dominated by highly frequent items during representation learning, as these items appear repeatedly in user interaction sequences. Their frequent occurrence leads them to contribute disproportionately to the optimization objective, biasing the learned representations toward popular items. As a result, existing models tend to emphasize popular items while failing to adequately capture low-frequency, long-tail items, even when such items may better reflect a user's underlying preferences or latent desires. This popularity-induced bias not only reduces recommendation diversity but also weakens the model's ability to accurately capture personalized user interests, particularly for users whose preferences are not well aligned with highly popular items or whose true intent is masked by frequent but less informative interactions.

Moreover, user interests naturally evolve across different temporal scales. Recent interactions are often indicative of transient and context-dependent short-term intents, capturing a user's immediate needs or situational preferences, while longer interaction histories tend to encode more stable and persistent long-term preferences. Previous studies have shown that jointly modeling these complementary interest signals can improve sequential recommendation performance, especially in scenarios involving complex or long user behavior sequences [5]. However, many existing approaches rely on relatively static fusion strategies, such as fixed weighting schemes or simple gating mechanisms, to combine long-term and short-term interests. Such designs offer limited flexibility in adjusting the relative importance of different interest components, making it difficult to adapt to dynamic changes in user behavior and evolving contextual conditions over time.

Beyond these considerations, most sequential recommendation frameworks integrate

multiple behavioral signals and interest representations in a largely deterministic manner. These designs often represent user behavior using fixed mappings or point estimates, which makes them less effective at capturing the uncertainty and variability inherent in real-world interaction data, especially in sparse or noisy settings. In such situations, the interactions observed in the data may only partially reveal a user's true preferences and are often affected by randomness, noise, or external influences in practice. Recent progress in generative modeling offers an alternative way of sequential recommendation [47]. Instead of relying on fixed point representations, these methods model user interests as evolving distributions through stochastic and iterative refinement processes, which allows them to better capture complex interest dynamics and to integrate diverse behavioral signals in a more adaptive manner.

Motivated by these observations, this thesis aims to investigate how sequential recommendation models can more effectively incorporate contextual behavioral signals, including temporal intervals and item popularity, while explicitly accounting for user interests at different temporal scales. In particular, the focus is on developing adaptive mechanisms that can flexibly integrate long-term preferences with short-term interests, with ability to better handle uncertainty and variability in real-world interaction data across diverse usage scenarios. The overall goal is to develop recommendation models that better capture the complexity of user behavior in practical settings, while enhancing robustness, flexibility, and overall recommendation performance.

1.3 Problem Statement

Although notable progress has been made in sequential recommendation, effectively modeling user behavior in real-world settings remains challenging. Existing methods often struggle to capture temporal irregularity, popularity imbalance, and the simultaneous presence of long-term preferences and short-term intents within a unified framework. This limitation makes it difficult for current models to accurately represent the dynamic nature of user interests and reduces their effectiveness in practical applications. To address these challenges, this thesis identifies several key modeling problems in sequential recommendation that motivate the development of more expressive and adaptive approaches.

Insufficient modeling of temporal behavioral signals: Many existing sequential recommendation models primarily rely on the order of user interactions while largely ignoring explicit temporal information, such as the time intervals between consecutive interactions. In real-world scenarios, however, user actions are often unevenly spaced in time, and the length of temporal gaps can carry important contextual information about user

intent and engagement patterns. By ignoring time interval signals, these models struggle to accurately represent variations in in user interaction frequency and intensity, and fail to capture the contextual meaning associated with different temporal gaps. This limitation restricts their ability to model dynamic user behavior and highlights the need for sequential recommendation approaches that explicitly incorporate temporal information into user behavior modeling.

Popularity-induced imbalance in sequential recommendation: Real-world interaction data is often dominated by a small number of highly frequent items, resulting in a highly skewed item distribution in user interaction sequences. Such imbalance can bias representation learning and recommendation outcomes, as popular items appear repeatedly and have a disproportionate impact during model training. Existing models tend to overemphasize popular items, while struggling to properly model low-frequency, long-tail items that may better reflect users' personalized interests. This popularity-induced bias not only limits recommendation diversity but also weakens the model's ability to capture individual-level user preferences. A key challenge in sequential recommendation is balancing the reduction of popularity bias with the preservation of meaningful user–item interaction patterns and accurate modeling of user behavior.

Limited expressiveness of deterministic long–short interest fusion: User interests naturally evolve across different temporal scales, where long-term preferences capture relatively stable and persistent tendencies, while short-term intents often reflect rapidly changing, context-dependent needs. Effectively modeling user behavior therefore requires capturing not only these interests individually, but also the complex interactions between them. However, many existing sequential recommendation models integrate long-term and short-term interests using deterministic fusion mechanisms, such as recurrent neural networks or simple linear combinations. These approaches typically perform interest fusion in a single step, producing a fixed representation that summarizes user interests at a given time. As a result, they lack sufficient expressiveness to model the detailed and evolving relationships between long-term preferences and short-term intents, limiting their ability to capture dynamic user interest transitions.

Lack of adaptive regulation in long–short interest integration: Beyond the choice of fusion paradigm, effectively combining long-term preferences and short-term intents also requires the ability to regulate their relative influence during the modeling process. In real-world scenarios, the importance of long-term and short-term interests may vary across time and contexts, and a fixed fusion outcome is often insufficient to reflect such variations. However, most existing methods either rely on fixed fusion outcomes or lack explicit mechanisms to control how the influence of long-term and short-term interests evolves during modeling. This limitation restricts the model's ability to maintain long-term

stability while incorporating short-term adaptability, highlighting the need for integration frameworks that can progressively and reliably modulate the influence of long-term and short-term interests.

In summary, this thesis addresses the problem of designing sequential recommendation models that are better suited to real-world user behavior. Specifically, it focuses on overcoming several key limitations of existing approaches, including insufficient modeling of temporal behavioral signals, bias introduced by highly skewed item popularity distributions, and limited expressiveness in integrating long-term preferences and short-term intents. These challenges make it difficult for current models to accurately represent dynamic user interests and to adapt to changing behavioral patterns and contextual conditions. This thesis aims to explore more expressive and adaptive modeling frameworks that can explicitly incorporate contextual behavioral signals, reduce the impact of popularity bias, and flexibly integrate user interests across different temporal scales, improving the robustness, personalization, and overall effectiveness of sequential recommender systems in practical settings.

1.4 Research Contributions

This thesis makes several contributions to the field of sequential recommendation by systematically addressing key limitations of existing methods in modeling contextual behavioral signals, popularity imbalance, and user interests at different temporal ranges. The main contributions are summarized as follows:

- **A context-aware framework for modeling dual-scale user interests in sequential recommendation.** This thesis proposes a unified modeling framework that explicitly incorporates contextual behavioral signals and user interests at different temporal ranges into sequential recommendation. By jointly considering temporal dynamics, item popularity effects, and the interaction between long-term preferences and short-term intents, the proposed framework provides a more expressive and realistic representation of evolving user behavior in real-world recommendation scenarios.
- **A time–frequency co-modeling approach for long-term and short-term interest representation.** To address insufficient modeling of temporal signals and popularity-induced bias, this thesis introduces a dual-gated time–frequency co-modeling strategy for sequential recommendation. Temporal intervals and item frequency information are explicitly integrated into both long-term and short-term interest modeling, enabling adaptive control over preference evolution while reducing

the influence of highly frequent items. This design improves the robustness of learned interest representations, particularly for capturing low-frequency and long-tail preferences.

- **An adaptive fusion mechanism for dual-scale interests based on diffusion-based generative modeling.** Moving beyond deterministic fusion strategies, this thesis proposes a novel interest integration approach based on diffusion-based generative modeling. Long-term preferences are treated as a stable generative prior, while short-term intents serve as conditional guidance during multi-step denoising. This progressive fusion process enables flexible and fine-grained interaction between long-term and short-term interests, allowing their relative contributions to be dynamically adjusted across the diffusion trajectory.

- **Contextual enhancement and controllable interest integration mechanisms.** To further improve the effectiveness and stability of long- and short-term interest fusion, this thesis introduces two complementary mechanisms. Token-level Contextual Enhancement captures fine-grained sequential cues from recent interactions through cross-attention, enriching short-term representations. In addition, a monotonic signal-to-noise ratio (SNR)–adaptive guidance mechanism is proposed to regulate the balance between long-term and short-term influences during diffusion, preventing noisy short-term behaviors from having excessive influence under high uncertainty and enabling a controlled transition toward short-term refinement.

- **Comprehensive evaluation on real-world benchmark datasets.** Extensive experiments are conducted on multiple real-world datasets from the Amazon Reviews corpus across different domains. The experimental results demonstrate that the proposed methods consistently outperform representative sequential recommendation baselines on multiple evaluation metrics, validating the effectiveness of context-aware modeling, popularity-aware representation learning, and adaptive long- and short-term interest fusion in improving recommendation accuracy, robustness, and personalization.

Overall, this thesis contributes a unified perspective on sequential recommendation that bridges contextual behavioral modeling, popularity-aware representation learning, and adaptive integration of dual-scale interests. The proposed approaches demonstrate how generative modeling can be effectively leveraged to move beyond deterministic fusion paradigms, enabling more flexible, interpretable, and robust modeling of evolving user interests in practical recommendation scenarios.

1.5 Publications

The work conducted in this thesis has resulted in the following publications:

- **Xiaowen, W., Tran, T. (2026).** "Dual-Gated Time–Frequency Co-Modeling for Sequential Recommendation." Submitted to User Modeling and User-Adapted Interaction. [Under Review]
- **Xiaowen, W., Tran, T. (2026).** "Diffusion-based Long and Short Term Interest Sequence Recommendation." Submitted to The 39th Canadian Conference on Artificial Intelligence. [Accepted]

These publications collectively represent the core contributions of this thesis. The first paper focuses on modeling temporal intervals and item frequency to enhance long- and short-term interest representation, while the second paper explores a diffusion-based generative framework for adaptive fusion of interests at different temporal ranges.

Chapter 2

Background Information, Literature Review, and Related Work

2.1 Introduction

This chapter aims to introduce the research of this thesis within existing work on recommender systems by clarifying the necessary background and organizing prior studies in a structured manner. Rather than introducing new models or results, the focus of this chapter is on establishing the context required to understand the problem setting, modeling choices, and assumptions underlying the methods proposed in later chapters.

The chapter is organized into three parts with distinct roles. Background Information (Section 2.2) presents the fundamental concepts and technical foundations of recommender systems that are relevant to this thesis. It introduces essential ideas and modeling components commonly used in sequential and hybrid recommendation, providing conceptual grounding without detailed comparison or critique.

Literature Review (Section 2.3) provides a structured overview of existing research. Prior work is grouped according to its main modeling strategy, which makes it possible to identify shared characteristics and limitations across different approaches.

Related Work (Section 2.4) examines studies that are most relevant to the models proposed in this thesis. These works are discussed in direct comparison to the proposed methods, emphasizing how they differ in their problem settings, input signals, and modeling mechanisms. This comparison helps situate the contributions of this thesis within the broader research landscape.

2.2 Background Information

2.2.1 Recommender Systems

Recommender systems are computational methods designed to suggest items that are likely to match a user's preferences based on available information. Given a set of users, a set of items, and historical interaction data between them, the goal of a recommender system is to predict which items a user may prefer or interact with in the future. Such systems are widely deployed across a range of online platforms, including e-commerce websites, streaming services, social media platforms, and online learning environments [37]. Typical recommendation targets include products, movies, music tracks, news articles, and other digital content. By providing user-specific recommendations, recommender systems aim to enhance user experience, increase engagement, and facilitate efficient access to relevant information. Personalization plays a central role in recommender systems. Rather than generating identical suggestions for all users, these systems model individual preference patterns and adjust their outputs accordingly. This emphasis on personalization distinguishes recommender systems from traditional information retrieval or search systems and motivates the use of user-specific representations [31].

The primary data source for recommender systems is user–item interaction data, which records how users interact with items over time. Such interactions may take various forms, including ratings, clicks, purchases, views, likes, or listening events. Each interaction is typically associated with a user identifier and an item identifier, and may additionally include side information such as timestamps or contextual attributes. User–item interactions are commonly categorized into explicit and implicit feedback [45]. Explicit feedback corresponds to direct expressions of user preference, such as numerical ratings or written reviews. Implicit feedback [12], in contrast, is inferred from observed user behavior, including clicks, views, purchases, or browsing activities, without requiring users to explicitly state their preferences. Interaction logs collected from real-world systems are typically organized as sequences with temporal ordering. They are typically sparse, unbalanced across users and items, and vary in length across users.

Different recommendation tasks depends on the application setting and system objectives. One common formulation is rating prediction, where the goal is to estimate a user's preference score for an item, typically represented as a numerical rating. Another widely used formulation is ranking-based recommendation, which focuses on ordering candidate items according to their relevance to a given user. In many practical systems, recommendation is defined as a top-k task. Given a user and a large pool of candidate items, the system produces a ranked list of K items that are most likely to be relevant or preferred [32]. This

formulation focuses on the relative ordering of items rather than the exact estimation of preference scores. In sequential recommendation scenarios, the task is further specified as next-item prediction. Based on a user's historical interaction sequence, the objective is to predict the next item the user will interact with. By explicitly modeling the order of interactions, this formulation relates recommendation tasks to sequence modeling.

Collaborative filtering (CF) is one of the most established paradigms in recommender systems. The main idea is to make recommendations by using patterns in historical user–item interactions, without depending on explicit item descriptions or user profile features. CF characterizes users and items implicitly through their observed behaviors, rather than predefined descriptive features [13]. CF methods are commonly divided into user-based and item-based approaches. User-based collaborative filtering recommends items favored by users who have exhibited similar interaction patterns. Item-based collaborative filtering instead focuses on similarities between items, suggesting items that are similar to those a user has previously interacted with. Both approaches build on the idea that similarity based on historical interactions can help indicate what a user may prefer in the future [13]. The collaborative filtering has influenced many subsequent recommendation techniques. Although later models introduce extra information or use more complex learning methods, many of them still follow the basic idea behind collaborative filtering.

Matrix factorization (MF) is a representative approach for implementing collaborative filtering through latent preference modeling [15]. In this approach, the user–item interaction matrix is factorized into low-dimensional vectors, with each user and item mapped to a point in the same latent space. These vectors capture preference patterns that are not directly visible in the raw interaction data. A predicted preference score is obtained by taking the inner product between the user vector and the item vector, which gives a simple way to quantify how well they match. This formulation offers a straightforward mechanism for modeling user–item relationships without requiring explicit feature descriptions. Matrix factorization can also be viewed as an early form of embedding-based recommendation. Once user and item vectors are treated as embeddings, later models extend the idea by adding neural architectures, extra behavioral signals, or other representation learning components, while still relying on the same basic structure.

Latent factor models give a compact way to represent user preferences, but they rely on aggregated interaction data and ignore the order of interactions. They mainly capture stable preference signals and do not show how a user's interest changes over time. This has led to approaches that learn vector representations for users and items and treat user behavior as an ordered sequence instead of a single profile. When the interaction order is included, the learned representations can reflect both long-term interests and more recent actions. This idea forms the basis for the representation learning and sequence-based

recommendation methods.

2.2.2 Representation Learning for Recommendation

Representation learning methods describe users and items using learned vector representations rather than predefined features. As the model learns from interactions, it builds a vector space where preference signals are encoded directly from the data. With users and items represented in a common vector space, the recommendation step reduces to evaluating how these vectors relate to one another through straightforward operations such as similarity scoring or distance measurement. By using learned embeddings, the model avoids depending on predefined feature structures or explicit content labels. The information extracted during training is stored directly in the embedding vectors, allowing the system to compare and rank large item sets efficiently without relying on manually created descriptors.

For user and item embeddings, each user and each item is assigned a learnable vector within the same space. These vectors are adjusted during training so that the model captures common patterns in user choices. As training progresses, users with comparable interaction histories are assigned vectors that fall close together, and items that appear in similar contexts for many users end up in neighboring areas of the latent space. Once the vectors have been learned, generating recommendations reduces to comparing a user's embedding with item embeddings through similarity or scoring functions. This approach keeps preference information in a compact form and allows fast retrieval, since the model only needs to perform simple vector operations.

By modeling the user behavior as an ordered sequence rather than a collection of unrelated events, the system produces a sequence-level representation instead of relying solely on separate item embeddings. Keeping the original interaction order allows the model to capture how the user's preferences develop across time. More recent behaviors can influence the representation more strongly, while earlier interactions still provide context for longer-term preferences. Depending on the application, the sequence representation may emphasize either extended histories or shorter, session-like behaviors. The resulting vector gives a snapshot of the user's current state based on both their past actions and the order in which those actions occurred.

After the user and item embeddings—or the sequence-level representation—have been learned, the model predicts user preferences through operations defined in the latent space. The model evaluates how well each item matches the user's current state by computing simple vector-based scores, which may involve inner products, similarity functions,

or lightweight transformations of the embeddings. These operations remain efficient even when the number of items is large, and the representations can be updated gradually as new interactions are observed. Since all computations rely on learned vectors rather than explicit item attributes or handcrafted features, the method offers a consistent way to reflect both long-term tendencies and short-term signals within the same framework.

2.2.3 Neural Sequence Modeling

Neural sequence modeling methods interpret user behavior as a time-ordered sequence of interactions and produce a representation that summarizes how the user's interests change across the sequence. Instead of treating past actions as isolated events, these methods process interactions in sequence, allowing the model to account for gradual changes in user preferences as new items appear. The final representation combines information from both earlier interactions and more recent actions, providing a clear view of the user's current behavior.

Recurrent neural networks (RNNs), including variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), process sequences by taking one interaction at a time and updating a hidden state that carries information from earlier steps. With each new input, the hidden state is adjusted to reflect both the previous state and the newly observed item, building a step-by-step summary of the user's behavior [3]. This hidden state is updated repeatedly as the sequence progresses, so the model can reflect how the user's behavior unfolds over time rather than treating earlier and later interactions as independent events. In recommender system, the user's interaction sequence is fed into the network step by step. Earlier actions influence the hidden state through accumulated updates, while more recent actions naturally have a stronger effect as they appear closer to the end of the sequence. This structure allows the model to represent both long-term behavior accumulated across many interactions and short-term interests driven by the most recent actions. LSTM and GRU extend the basic RNN structure by adding gating components that regulate how much past information is kept or replaced. These gates help maintain stability when processing long sequences and allow the final hidden state to represent both persistent behavioral patterns and momentary preferences without explicitly storing the entire sequence.

Attention-based models offer an alternative way to represent user interaction sequences by allowing the model to directly reference any earlier interaction when forming the current representation. Instead of compressing the sequence into a single hidden state, the attention mechanism evaluates all past interactions in parallel and assigns different weights to each,

depending on how useful they are for the current prediction. This makes it possible for the model to emphasize interactions that carry strong signals and reduce the impact of those that contribute less, while avoiding the constraints of a strictly step-by-step update process. The idea behind attention is that interactions in a sequence do not all influence future behavior to the same extent. Some items provide clearer indications of a user's next choice, while others are only weakly related. By computing a relevance score for each interaction and turning these scores into weights, attention produces a representation that highlights sequence elements that matter most. Because the mechanism can directly compare all positions, it captures long-range dependencies that would otherwise fade when information is passed forward through many sequential updates [35]. A common formulation uses Query, Key, and Value vectors. The current position is represented by a Query, and each past interaction produces a Key and a Value. The Query is compared with all Keys to compute attention weights, and these weights are applied to the Values to form a weighted combination of relevant interactions. This result becomes the sequence representation used for the next prediction step. By constructing the representation in this way, the model incorporates information from the entire sequence without depending on a single chain of hidden states.

2.2.4 Transformer-based Sequence Modeling

Transformer-based sequence modeling represents user behavior using self-attention rather than step-by-step updates. In this framework, all interactions in the sequence are processed in parallel, and each position can attend to any other position directly. This allows the model to capture relationships across the entire sequence when forming the final representation, without relying on a single hidden state that is passed through the sequence.

Self-attention computes how strongly each interaction in the sequence relates to every other interaction. For each position, the model forms a Query vector that is compared with Key vectors from all positions in the sequence. The resulting similarity scores determine how much influence each past interaction should have. These scores are then used to compute a weighted combination of Value vectors associated with each position. Through this process, the representation at each position is formed by gathering information from the remaining positions in the sequence. Interactions that are more relevant receive higher weights and contribute more strongly, while unrelated interactions contribute less. By allowing every position to directly reference any other position, self-attention captures both local patterns and distant dependencies without being restricted by sequential update order [35].

Since self-attention views all positions at the same time, it dose not automatically encode the order in which interactions occur. To incorporate temporal structure, positional encoding is added to the input embeddings. This encoding provides numerical signals that identify the position of each interaction within the sequence and allow the model to distinguish earlier interactions from later ones. Positional information can be added through deterministic functions or learned vectors, and it ensures that the final representation reflects not only which items occurred but also when they occurred.

Transformer-based modeling differs from recurrent models in how sequence information flows. RNN-style models read the sequence one step at a time and update a hidden state recursively. Information about earlier interactions must be carried forward through the hidden state, so distant dependencies may weaken as the sequence grows longer. Transformer-based models, in contrast, allow every position to access the entire sequence directly through self-attention. The representation does not rely on incremental updates, and interactions that are far apart can still influence each other through attention weights. This structural difference leads to a representation that can integrate information from multiple parts of the sequence at once, while still retaining temporal order through positional encoding.

2.2.5 User Interest Modeling at Different Temporal Ranges

User behavior in interactive systems often reflects preferences that operate over different temporal ranges. Some patterns remain stable across long periods and represent broad interests, while others change rapidly and correspond to short-term or session-specific intentions. Modeling user interest across different temporal ranges provides a way to represent both stable patterns and recent signals within the same framework, allowing the system to account for how different forms of behavior influence recommendations.

Long-term preferences typically develop over many interactions and remain consistent across extended usage. These preferences may correspond to general categories, familiar brands, or stable content preferences that continue to influence the user's choices. Since they change slowly, long-term patterns provide reliable context for understanding the user's overall behavior. In contrast, short-term intents reflect the user's immediate goals or current focus. These signals are often shaped by recent interactions and can shift quickly depending on the user's context, such as browsing within a specific product subcategory or following a temporary interest trend. Short-term behavior usually plays a significant role in determining what the user is likely to do next, even when it differs from their broader preferences.

These two temporal scales contribute differently to how users make decisions. Long-term preferences define the broader scope of items the user tends to favor and influence choices that align with recurring patterns. They provide background structure and continuity across sessions. Short-term intents affect decisions that occur closer in time to the most recent interactions. They capture immediate signals that can guide the next step more directly, such as refining a search focus or narrowing interest within a specific category. While long-term preferences provide stability, short-term intents introduce flexibility and allow the system to adapt to immediate shifts in the user's behavior.

Since long-term preferences and short-term intents influence user behavior in different ways, representing them within a single temporal scale can constrain how the system interprets ongoing activity. A dual-scale approach allows the model to handle stable patterns and recent signals separately, rather than forcing them into a single form. This leads to a representation that captures both lasting interests and immediate context, providing a more balanced view of the user's state. By forming dedicated components for different temporal ranges and merging them when producing recommendations, dual-scale modeling offers a structured way to describe user behavior. Modeling these temporal aspects separately can reflect how user interests change over time and provide a foundation for sequence-aware methods that incorporate information from different points in the user's history.

2.3 Literature Review

Recommender systems have evolved substantially alongside the increasing scale and complexity of user–item interactions in modern digital platforms. Early approaches relied primarily on collaborative filtering and content-based strategies [44], which leveraged historical preferences or item attributes to generate recommendations. While effective in structured environments, these methods struggled to capture sequential dependencies, temporal variation, and the dynamic nature of user behavior. As digital services began recording more detailed interaction histories with timestamps, sequential recommendation became an important direction, shifting the focus from static preference modeling toward understanding how user interests evolve over time. This shift reflects the need to model not only what users interacted with, but also when and in what order those interactions occurred. As user behavior became more continuous and fine-grained, traditional static models increasingly showed their limitations, motivating the development of methods explicitly designed for dynamic and temporally dependent interaction patterns.

One challenge for sequential modeling is how to incorporate temporal information in

a meaningful and reliable way. User actions occur at irregular intervals, and the duration between interactions often contain information about recency, decay of interest, and intention to change. Methods that treat interaction sequences as uniformly spaced overlook these temporal dynamics [17, 46] and may ignore the relative importance of past behaviors over time. Later research has emphasized the importance of time-aware modeling, exploring mechanisms to encode time gaps, model temporal decay, or adjust attention based on temporal distance. These approaches improve the ability to represent evolving interests but often address temporal signals independently, without considering other influential behavioral factors such as item popularity or multi-scale preference structure. Time-aware components are commonly used as add-on features rather than being integrated into the core representation learning process, limiting their overall impact.

Another important challenge arises from item popularity and frequency imbalance. Real-world interaction data commonly follow a long-tailed distribution [21] in which a small number of popular items appear frequently while many others remain sparse. Sequential models that rely heavily on gradient-driven learning or attention mechanisms can unintentionally focus on frequent items [14] and under-represent the long tail. This imbalance reduces recommendation diversity and limits the system's capacity to serve users with less mainstream preferences. Prior studies explored reweighting strategies, regularization terms, and representation enhancement techniques to mitigate this bias [22], while these solutions often operate separately from the core sequence encoder. Popularity information remains under-utilized, and temporal dynamics are rarely integrated with frequency signals in a unified modeling process.

In addition to temporal and popularity considerations, user interests typically span multiple timescales. Long-term preferences characterize general tastes that persist for extended periods, while short-term intents reflect immediate goals or session-specific motivations [20]. Treating user behavior as a single uniform sequence can ignore this multi-scale structure [43] and cause recent but important actions to be overwhelmed by historical patterns. Research in this direction has highlighted the importance of separating long-term and short-term interests and designing mechanisms to combine them effectively [23, 26, 33]. Existing studies have adopted hierarchical, attention-based, or memory-augmented frameworks to capture interest signals at different temporal levels. Although these methods improve the representation of evolving behavior, many still rely on static or single-step fusion strategies [6] limiting their ability to adaptively balance long-term stability and short-term specificity under varying user contexts.

Beyond deterministic sequence modeling, generative approaches have drawn increasing attention because they can express uncertainty and represent multiple possible user intentions rather than committing to a single prediction. Generative architectures—such as

variational latent models [19], adversarial learning frameworks [36], and iterative refinement mechanisms—offer the ability to represent preference distributions rather than single-point predictions. In particular, diffusion-based generative models have recently demonstrated strong potential for sequential modeling [38] by transforming prediction into a multi-stage denoising process. Their iterative nature allows gradual refinement of interest representations and provides opportunities for conditioning on auxiliary behavioral signals. However, current generative approaches typically model sequences as a whole [16,18] and lack explicit mechanisms for controlling or disentangling long- and short-term interest components.

Overall, prior research has addressed temporal dynamics, popularity imbalance, interest multi-scale structure, and generative modeling—but largely as separate lines of investigation. Existing systems tend to specialize in one or two dimensions, such as handling temporal decay or reducing popularity bias, without providing a unified solution capable of jointly modeling time intervals, frequency patterns, and long–short interest interactions. This fragmentation highlights an opportunity for approaches that integrate these complementary signals within a coherent framework, enabling more adaptive, interpretable, and fine-grained sequential recommendation.

2.4 Related Work

2.4.1 Introduction

This section reviews the research directions that form the foundation of modern sequential recommendation and are most relevant to the scope of this thesis. The discussion highlights representative modeling paradigms, their core assumptions, and the challenges they seek to address. To provide a structured overview, the following subsections examine five major categories: general sequential recommendation models, time-aware extensions, popularity-aware techniques, long–short interest modeling approaches, and generative frameworks. Overall, these perspectives illustrate the evolution of sequential recommendation from traditional deterministic architectures to adaptive, temporally sensitive, and distribution-aware modeling strategies. By outlining their strengths and limitations, this section establishes the context in which the subsequent methodology is developed and evaluated.

2.4.2 Sequential Recommendation Models

Recommender systems are an essential component of modern e-commerce platforms, enabling users to efficiently discover relevant items within diverse catalogs. By analyzing users' historical interactions, recommender systems can infer implicit preferences and deliver personalized recommendations that improve user satisfaction. However, user preferences are not static. They evolve over time as users face with new contexts and content. As a result, effectively modeling these dynamic interests from sequential user behavior has become a challenge in improving recommendation quality.

Traditional recommendation techniques mainly include content-based recommendation and collaborative filtering (CF) methods [44]. Early CF models decompose the user–item rating matrix to learn latent representations and recommend items with similar embedding patterns. Matrix Factorization (MF) [1] is a representative example, which computes the inner product between user and item embeddings to estimate preference scores. However, these methods ignore the order of interactions and therefore cannot capture how user interests change over time. Moreover, such models assume static user preferences and fail to account for behavioral recency, making them unsuitable for scenarios where user interests evolve quickly.

Sequential recommendation systems extend these ideas by incorporating the temporal order of user interactions. They aim to model sequential user behaviors, user–item interactions, and the evolution of user preferences and item popularity over time [37]. Sequential recommendation predicts a user's future interactions by modeling the temporal dependencies within their behavioral sequences.

Early studies focus on long-term preferences through Markov-based models. FPMC [30] combines matrix factorization with personalized Markov chains to jointly model users' global preferences and their most recent interaction. Fossil [6] further extends to higher-order Markov chains and integrates item similarity to capture more complex sequential transitions. With the growth of behavior data, research gradually shifted toward modeling short-term interests to better reflect rapidly changing user intents. These Markov-based approaches are simple and efficient, and they effectively capture immediate transitions between items. However, their reliance on predefined transition orders limits their ability to model complex interaction patterns that depend on broader historical context.

To capture patterns beyond simple first-order transitions, neural models were later introduced. Recurrent Neural Network (RNN) based models such as GRU4Rec [9] and GRU4Rec+ [8] provide a neural approach to capturing temporal dependencies, updating hidden states step by step to represent evolving user interests. Caser [34] adopts convolution architectures to extract local sequential patterns for next-item prediction. Compared

with Markov-based methods, RNN architectures are better suited for capturing sequential dependencies of variable lengths and can model richer transition dynamics through learned hidden representations. While RNN-based methods are effective in modeling short-term dependencies, they face several limitations. Gradient vanishing and explosion problems limit their ability to handle long interaction histories, and their local nature makes it difficult to efficiently capture global dependencies [24]. In addition, their sequential computation structure prevents parallelization during training, resulting in substantially higher computational cost for long sequences. These challenges have led researchers to explore alternative architectures that can model long-range interactions more flexibly and effectively.

Despite their effectiveness, RNN-based methods encounter computational and modeling limitations when sequences grow long. To overcome these problem, self-attention mechanisms have been introduced into sequential recommendation. MARank [43] leverages multi-attention to capture user interests at multiple dependency levels, and SASRec [14] applies self-attention to model user sequences globally. These approaches allow each item in a sequence to attend to all others, enabling flexible modeling of long-range dependencies with high parallelism. This formulation enables direct access to global context and allows the model to focus selectively on behavior patterns that are most relevant for prediction, improving expressive power and interpretability. TiSASRec [17] further incorporates time-aware attention to refine temporal modeling. Compared with RNNs, attention-based models remove the need for step-by-step updates and therefore handle long-range relationships more efficiently. However, standard attention-based models treat interactions as uniformly spaced in time and therefore cannot distinguish between recent actions and older ones unless additional temporal signals are incorporated.

2.4.3 Time-aware Sequential Recommendation

Modeling temporal signals is crucial in sequential recommendation because user behaviors are inherently time-dependent. The time interval between two interactions contains rich information about behavioral recency, interest decay, and contextual relevance. For example, actions occurring within a short period often share the same intent, whereas long gaps between interactions may indicate a shift in preference. However, traditional sequential models typically rely on ordered sequences without explicitly encoding the magnitude of time intervals, limiting their ability to differentiate stable long-term preferences from transient short-term motivations. While this simplified formulation allows efficient computation and reduces model complexity, it also ignores important temporal cues such as

recency and behavioral drift, which reduces predictive accuracy in scenarios with irregular or sparse timestamps.

Early neural sequence models, such as LSTM [28], partially alleviate the vanishing-gradient problems of standard RNNs through gating mechanisms, but they still treat all neighboring actions as uniformly spaced in time. To address this limitation, Time-LSTM [46] incorporates time gates that explicitly incorporate elapsed time into the state update. The time gate dynamically regulates the contribution of recent actions based on the interval length, allowing the model to discount outdated behaviors and emphasize fresh signals. By integrating both item transitions and temporal intervals into the memory cell, Time-LSTM effectively captures both short-term interests and stable long-term patterns, improving accuracy in scenarios where interaction timestamps are irregular. Compared with standard LSTM variants, this design provides a more principled way to regulate temporal decay and adapt to time-heterogeneous interaction sequences. However, Time-LSTM still inherits the sequential recurrence structure of RNNs, which limits its parallelization capability and makes it computationally expensive for very long sequences.

Beyond recurrent networks, time-aware modeling has also been extended to attention-based architectures. SASRec [14] demonstrates that self-attention can flexibly capture long-range dependencies by allowing each item to attend to all previous items with adaptive weights. However, SASRec does not encode real-valued temporal intervals, causing items with the same position distance but very different real-world time gaps to be treated similarly. This limitation can cause the model to overestimate the relevance of outdated interactions, particularly in domains where user interests shift quickly or timestamps are highly irregular. This motivated the development of TiSASRec [17], which introduces learnable time embeddings and time-aware attention biases to explicitly inject temporal distance into the self-attention mechanism. By jointly modeling item-to-item and time-to-time relations, TiSASRec enables the representation to reflect finer-grained temporal patterns and better distinguish between persistent preferences and rapidly shifting short-term intents. This time-aware formulation enhances the model's ability to capture recency effects and temporal proximity, improving next-item prediction especially in domains with strong short-term behavioral signals.

Although these methods successfully incorporate temporal distance into sequence modeling and improve temporal sensitivity, they still rely on representing user behavior with a single unified embedding. Most time-aware models—including Time-LSTM and TiSASRec—still follow a single-interest representation paradigm, encoding the entire user history into one unified embedding. This design ignores the structural distinction between stable long-term preferences and short-lived short-term motivations, limiting flexibility in scenarios where user intentions evolve quickly or where recommendations require balancing histor-

ical tendencies with immediate contextual cues. These limitations further highlight the need for multi-scale modeling strategies capable of jointly leveraging behavioral signals across heterogeneous temporal resolutions.

2.4.4 Popularity-aware Methods

In sequential recommendation, item popularity has an effect on how models learn behavioral patterns and generate predictions. Since user interactions are typically dominated by a small subset of highly popular items, many sequence encoders tend to prioritize these frequent patterns while overlooking rare or long-tail items. This imbalance leads to popularity bias, which reduces recommendation diversity and undermines the system's ability to capture users whose interests involve less common categories. Moreover, sequential models that rely on recurrent or self-attention mechanisms often reinforce this imbalance, since frequently occurring items accumulate stronger gradients and are assigned relatively high attention weights during training. This issue becomes more pronounced in datasets with skewed interaction distributions, where popular items dominate gradient updates and suppress informative signals from tail items.

To address this issue, recent popularity-aware approaches incorporate frequency or popularity signals directly into the representation learning process, rather than treating them as external correction terms. UniRec [22] is a representative example of this direction. It introduces a dual-enhancement framework that simultaneously encourages uniform attention across sequence positions and strengthens the representations of low-frequency items. By explicitly enhancing tail-item embeddings and smoothing position-wise attention, UniRec reduces the model's dependence on high-frequency patterns and improves robustness on sparse or irregular sequences. This unified treatment allows the sequence encoder to better represent long-tail items and improves generalization across datasets. A key advantage of this framework is that it enhances tail-item visibility directly within the encoder, avoiding the need for post-hoc reweighting or auxiliary sampling strategies. However, UniRec does not explicitly account for temporal decay or contextual dynamics, making it less effective when user preferences shift rapidly over time.

Another line of research strengthens tail-item modeling through specialized memory components or retrieval-enhanced architectures. MASR [11] follows this direction by introducing a memory-bank-based retriever–copy framework. Instead of depending solely on the internal sequence encoder, MASR constructs a cluster-wise memory bank that stores historical representations of low-frequency items and retrieves similar behavioral patterns during inference. The retrieved features are then injected into the prediction process to

reinforce signals associated with underrepresented items. This retrieval-based design allows the model to use historical patterns even when direct interactions with tail items are extremely sparse. This mechanism provides additional support for tail items that appear infrequently during training. However, MASR does not explicitly model temporal decay or time-interval dependencies, which limits its ability to capture how the relevance of tail items evolves over time. In addition, the external memory structure introduces computational and storage overhead during inference, limiting scalability in large-scale real-world environments.

Frequency-aware modeling has also been explored from the perspective of analyzing how sequence encoders behave in the frequency domain. FEARec [4] introduces that conventional self-attention architectures function similarly to low-pass filters, reducing the influence of high-frequency components within user interaction sequences. To address this limitation, FEARec introduces frequency-domain auto-correlation modules and jointly learns representations in both the time and frequency domains. This dual-view mechanism enables the model to recognize repeated behavioral cycles and maintain detailed variations in user interactions—patterns that standard attention layers often smooth out. By combining information from both the time domain and the frequency domain, the method strengthens the representation of less frequent or irregular user behaviors, all without relying on extra sampling tricks or customized loss reweighting. This provides a frequency-domain perspective that complements traditional time-domain modeling and offers improved interpretability of periodic behavioral patterns. While FEARec focuses on capturing frequency features and does not explicitly integrate popularity statistics or temporal intervals into a unified representation, limiting its ability to respond to dynamic user preference shifts.

Overall, popularity-aware and frequency-aware techniques aim to counteract the dominance of frequent items and improve the handling of long-tail interactions. Existing solutions enhance tail-item representations, introduce memory-based retrieval, or adjust attention mechanisms to preserve high-frequency patterns. However, these approaches usually model popularity and temporal information independently rather than in a coordinated way. This reveals an opportunity for frameworks that integrate frequency, time-interval signals, and sequential dependencies into a single, adaptive representation.

2.4.5 Long–Short Interest Modeling

While single-interest sequence modeling is effective in many scenarios, it struggles to capture the complementary roles of a user's stable long-term preferences and rapidly changing short-term intents. To address this limitation, a variety of long–short interest modeling

approaches have been introduced, each aiming to explicitly separate and fuse information from different temporal ranges.

Early work incorporated long-term and short-term signals using static or learnable weights. Fossil [6], for example, integrates collaborative filtering components for long-term preference with higher-order Markov transitions for short-term dynamics, but its linear fusion mechanism remains fixed and lacks contextual adaptability. Similarly, TDSSM [33] models long-term and short-term representations through neural architectures but still combines them through relatively static integration strategies, limiting its ability to capture rapid interest shifts. A notable strength of these early approaches is their conceptual simplicity, which keeps computation efficient and avoids architectural overhead. However, this simplicity limits their expressiveness when user behavior becomes highly dynamic or context-dependent.

Subsequent research introduced gating mechanisms to dynamically regulate the influence of each interest component. The Hierarchical Gating Network (HGN) [23] develops parallel long-term and short-term channels, using an item-aware gate to adaptively adjust their contribution based on contextual relevance. This gating formulation offers greater flexibility than static fusion but still merges only two fixed interest vectors at a single point in the network, limiting deeper interaction between temporal components. In addition, the gating decision is made at a single model layer, which restricts its ability to refine the balance between interests across multiple stages of reasoning.

Attention-based models capture multi-scale behavior through dependency-aware weighting. MARank [43] employs multi-order attention to highlight both immediate and distant dependencies, explicitly extracting long-term and short-term interest signals. STAMP [20] focuses on session-based short-term intent by combining an attention-derived session context with the most recent click. These attention-based models offer strong interpretability by revealing which historical actions contribute to short-term or long-term preferences. Although these methods allow more expressive interest extraction, they ultimately fuse long-term and short-term representations through a single-step attention mechanism that does not dynamically evolve over multiple stages of inference.

More advanced approaches maintain persistent long-term patterns using external memory structures. MIMN [26] stores multiple latent long-term interests in memory slots and retrieves them using the current short-term context as a query. Its induction unit updates memory states with new behaviors, enabling a form of dynamic fusion across time. However, memory-augmented models often incur substantial computational and storage overhead, and the fusion between long- and short-term interests remains implicit rather than explicitly controllable. Moreover, updating and retrieving from memory can introduce

latency during inference, making these models less practical for real-time recommendation systems.

Long–short interest modeling methods share common characteristics. They separate long-term and short-term signals but typically combine them through static fusion, single-step gating, or attention-based aggregation. These designs limit their ability to continuously regulate the contribution of each temporal component and to adapt to evolving user intents, motivating the need for more flexible and progressive fusion frameworks. These approaches demonstrate that explicitly modeling multiple temporal ranges consistently improves recommendation accuracy compared with single-interest architectures.

2.4.6 Generative Models for Recommendation

Generative modeling provides an alternative perspective for recommendation by viewing user–item interactions as samples drawn from an underlying distribution rather than deterministic sequences. Early generative approaches include Variational AutoEncoder (VAE)-based and Generative Adversarial Networks (GAN)-based recommendation frameworks, which aim to capture the stochasticity and uncertainty that naturally arise in user behaviors. VAE-based methods model latent preference distributions through probabilistic encoders and decoders, enabling better generalization under sparse data [19]. GAN-based recommenders, including IRGAN [36], employ adversarial training to generate item embeddings or user preference vectors that match the distribution of real interactions. These generative paradigms highlight the importance of distributional modeling for representing uncertainty and multi-modal user interests. A major strength of these models is their ability to capture diverse preference patterns that are difficult to model with deterministic encoders, especially when user interactions produce noise.

Building on these insights, diffusion models have recently gained attention in sequential recommendation. Diffusion-based generative models operate through a multi-step denoising process, gradually transforming random noise into structured representations conditioned on user interaction histories. DiffRec [38] is one of the earliest diffusion-based sequential recommenders, formulating next-item prediction as a generative refinement process. By progressively corrupting and reconstructing item embeddings, DiffRec effectively captures dependency structures within user sequences and demonstrates notable performance gains over traditional deterministic models.

Following DiffRec, subsequent works extend diffusion frameworks to improve the expressiveness and stability of generative recommendation. DiffuRec [18] models user–item

interactions as probability distributions and uses diffusion to capture uncertainty and multi-interest patterns more explicitly. EDiffuRec [16] further enhances the generative process by incorporating anoise distribution, a strengthened Transformer backbone, and training techniques such as normalized loss scaling and warmup scheduling to improve convergence and robustness. These improvements demonstrate the adaptability of diffusion-based models and their capacity to handle complex sequential patterns.

While the majority of current diffusion-based approaches are optimized for reconstruction quality rather than explicit temporal reasoning, which limits their ability to control how different behavioral signals contribute across denoising steps. Existing diffusion-based recommenders primarily focus on sequence reconstruction or noise scheduling and often lack explicit mechanisms for modeling multi-scale user preferences. Most frameworks treat the entire interaction sequence as a unified representation, without explicitly disentangling long-term preference signals from short-term contextual cues. As a result, they may struggle to regulate the relative contributions of stable historical interests and rapidly changing recent behaviors during the denoising process. This limitation underscores the need for generative models that incorporate structured long–short interest modeling and adaptive temporal guidance within the diffusion trajectory.

Chapter 3

Dual-Gated Time–Frequency Co-Modeling for Sequential Recommendation

3.1 Introduction

In this chapter, we present the development and analysis of a dual-gated time- and frequency-aware sequential recommendation model, referred to as **TiIfSRec**. The model is designed to address key challenges in real-world user behavior data, where temporal irregularity, interest drift, and long-tail sparsity jointly affect recommendation accuracy. Rather than relying solely on raw interaction sequences, TiIfSRec explicitly incorporates time intervals and item frequency information as complementary behavioral signals to enhance the representation of evolving user preferences.

Traditional sequential recommender models typically rely on recurrent or attention-based architectures to capture dependency patterns within user histories. While effective in many settings, these approaches tend to treat interactions as evenly spaced in time and frequently over-emphasize popular items. Consequently, short-term intents may be overshadowed by outdated behaviors, and long-tail items may receive insufficient representation during training. These limitations are especially evident in domains with irregular user activity or sparse interaction sequences.

TiIfSRec addresses these issues through a dual-gated GRU module that integrates temporal distance and item frequency into the hidden-state update process. The time gate

adaptively modulates how strongly past interactions influence current predictions based on elapsed time, allowing the model to discount stale information while highlighting recent behavior. Meanwhile, the frequency gate counteracts popularity imbalance by strengthening low-frequency item signals and reducing excessive bias toward frequent items. Together, these mechanisms enable the model to simultaneously capture persistent long-term preferences and rapidly shifting short-term intents within a unified architecture.

This chapter makes several key contributions:

- It introduces a unified formulation that jointly models time intervals and item frequency within a gated recurrent architecture, enabling more nuanced sequential representations.
- It proposes a dual-gated mechanism that separately regulates temporal decay and popularity bias, improving the balance between long-term stability and short-term specificity.
- It demonstrates empirical effectiveness across multiple benchmark datasets, showing consistent gains over GRU-based and attention-based baselines under various data sparsity and temporal irregularity conditions.

The remainder of this chapter is organized as follows. Section 3.2 presents the theoretical foundations and architectural components of TiIfSRec, including the long-time module with time-gate and frequency-gate and short-time module. Section 3.3 describes the experimental setup, datasets, baseline models, and evaluation metrics, and reports and analyzes the experimental results. Section 3.4 discusses the implications, limitations, and practical considerations of the proposed model, and summarizes the chapter and prepares the foundation for the advanced generative modeling approach introduced in the next chapter.

3.2 Methodology

3.2.1 Architecture

The proposed TiIfSRec model aims to effectively capture both short-term and long-term user interests by jointly modeling time intervals and item frequencies. As illustrated in Figure 3.1, TiIfSRec consists of three main components: a long-term interest module, a short-term interest module, and a fusion gate that integrates the two representations into

a unified user embedding for recommendation. The long-term interest module is in Section 3.2.4, and the short-term module in Section 3.2.5. T_t and F_t are time gate and frequency gate in Section 3.2.6. i'_t denotes the long-term item embedding at time step t, and f_t is the long-term item frequency embedding at time step t. Att denotes the self-attention weight in Equation 3.15 3.16. Q, K, and V are defined in Equation 3.23 3.24 3.25. r_t, z_t, and h_t are reset gate, update gate, and hidden state, respectively. Activation functions sigmoid σ, tanh, and Softmax are defined in Section 3.2.4.

In the long-term interest module, the GRU architecture is enhanced with a dual-gated mechanism that jointly models temporal decay and frequency-driven directional modulation. The time and frequency gates are fused to produce an adaptive decay rate that determines how much past preference should be attenuated over time. A frequency-controlled directional gate adjusts the candidate hidden state's direction by down-weighting the dominance of highly frequent items and preserving informative low-frequency signals. This design enables selective memory retention, adaptive preference shifts, and effective mitigation of popularity bias while maintaining temporal consistency in long-term user modeling. In the short-term interest module, a GRU encodes fine-grained time intervals and implicit positional information, and a self-attention mechanism is applied to integrate frequency embeddings, allowing the model to focus on the most recent and informative interactions. Finally, a fusion gate dynamically combines the outputs of both modules to generate a comprehensive representation of user interest that balances long-term stability with short-term adaptability.

3.2.2 Embedding

To make the user interaction data accessible for modeling, the embedding layer maps raw inputs such as user, item, and temporal information into a low-dimensional space while preserving their latent patterns and correlations.

Let U be a set of users and I be a set of items. Let u, i, t represent user ID, item ID, and interaction timestamp, respectively. Let L be the length of the interaction sequence. For each user $u \in U$, the interaction sequence of the user can be formulated as a sequence of items ordered by timestamps, where the item sequence is $\{i_1, \ldots, i_L\}$, and the timestamp sequence is $\{t_1, \ldots, t_L\}$.

For each item $i \in I$, the title content of the item can be formulated as c^i. The training sequence for long-term interest is $I_{\text{long}} = \{i_1, \ldots, i_{L-2}\}$. For short-term interest, the training sequence is transformed into a sequence of fixed length n, so we have $I_{\text{short}} = \{i_{L-2-n}, \ldots, i_{L-2}\}$. If the sequence length is greater than or equal to n, only

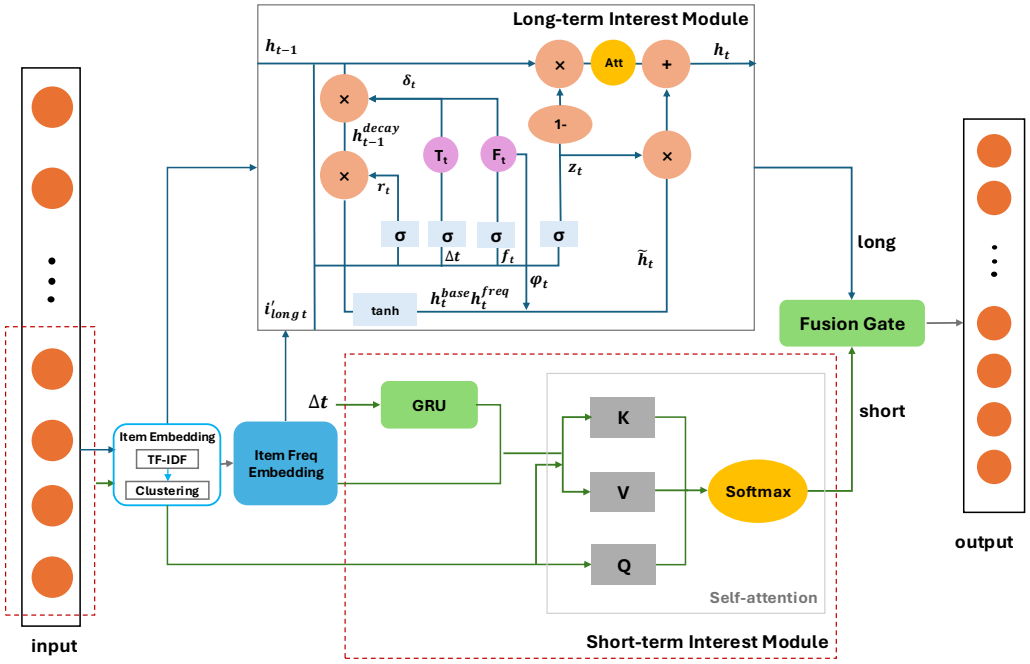


Figure 3.1: The overview framework of TiIfSRec model

the most recent n interactive items are considered. If the length of the sequence is less than n, the padding item is added to the left until the length is n. Similarly, the time sequence of long-term interest is $\{t_1, \ldots, t_{L-2}\}$, and the time sequence of short-term interest is $\{t_{L-2-n}, \ldots, t_{L-2}\}$. The time intervals are the differences between two timestamps, and the first interval is assigned the value of the previous interval to avoid zero. Time intervals are $\{|t_1 - t_2|, \ldots, |t_{L-2} - t_{L-1}|\}$ for long-term interest, and $\{|t_{L-2-n} - t_{L-2-n+1}|, \ldots, |t_{L-2} - t_{L-1}|\}$ for short-term interest. The time intervals are then scaled by Min-Max scaler and transformed into the range $[0, 1]$.

$$\Delta t' = \frac{\Delta t - \Delta t_{\min}}{\Delta t_{\max} - \Delta t_{\min}} \tag{3.1}$$

The training target is to predict the item at t_{L-1}, and the testing target is to predict the item at t_L.

3.2.3 Unsupervised learning

In practice, the number of items is usually very large, while the number of items each user interacts with is very limited, and these interacted items usually have great diversity. Even the items under the same category may have thousands of varieties, for example, a dress can be categorized by different attributes such as brand, color, material, and size, etc. Hence, it is difficult for the model to directly use the item IDs as inputs, and the model's lack of sufficient historical data can lead to a data sparsity problem, which affects the recommendation performance. Replacing item IDs with broader categories may reduce sparsity but could lose essential information, and thus reduces the accuracy of the recommendations. Therefore, the data sparsity problem is solved by using unsupervised learning, clustering by learning the titles of items to generate new item classes to replace item IDs.

Each item has its own specific title to help users quickly understand what the item is about when browsing. Titles usually have a brief description of the item, including its usage, brand, size, function, target group, etc. In this work, the title refers to the product title field provided in the dataset, which typically contains concise semantic cues such as keywords, category indicators, or series information (e.g., movie titles may include themes, genres, or series names), rather than user reviews or detailed descriptions. Using item title information instead of item IDs reduces data sparsity issues. Term Frequency-Inverse Document Frequency (TF-IDF) [29] is a text feature extraction method that can highlight words that are more important to the topic and ignores words that occur frequently but are not distinguishable, such as "the", "is", etc. Instead of modeling semantic relationships explicitly, this approach groups items based on the co-occurrence of informative keywords in their titles. Items that share overlapping keywords will have similar TF-IDF representations and are therefore more likely to be assigned to the same cluster. In this way, user interactions are aggregated at the cluster level, increasing interaction overlap and alleviating data sparsity. Item titles are processed by TF-IDF method to extract main features. Each TF-IDF representation is a vector of dimension, which is the size of the vocabulary constructed from all item titles.

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t) \tag{3.2}$$

where TF denotes the frequency of occurrence of a word (term) t in title d, and IDF denotes Inverse Document Frequency to reduce the weight of common terms.

To solve the problem of reducing data sparsity, the words extracted from the item title are clustered using the unsupervised learning method KMeans. Equation 3.3 defines the objective function of the KMeans clustering algorithm, which minimizes the total distance

between each item's TF-IDF vector and their cluster centroid vector. This process assigns the same label to items with similar extracted features, which are then used as new item IDs I'. By sharing these cluster-based IDs across semantically similar items, user behaviors become more comparable, which increases interaction overlap and reduces data sparsity.

$$J = \sum_{c=1}^{k} \sum_{i'_j \in S_c} \|i'_j - \mu_c\|^2 \tag{3.3}$$

where i'_j denotes the TF-IDF vector extracted from the title of the j-th item assigned to cluster c, μ_c denotes the centroid of cluster c, and S_c is the set of vectors in cluster c. k denotes the number of clusters, which is selected empirically based on validation performance. $\|\cdot\|$ is the Euclidean norm (i.e., vector length). J is the cost function to find the best clustering.

After obtaining cluster assignments, raw item IDs are replaced with their corresponding cluster IDs and train the recommendation model directly on these cluster-level representations. This transformation converts the original sparse item space into a denser cluster space, where items with similar semantic content share the same identifier. Inference is therefore performed at the cluster level, and the predicted cluster serves as the compressed representation of the next item. This approach reduces sparsity while preserving semantic proximity among items within the same cluster, without requiring additional item-level ranking inside each cluster. The generated long-term item IDs embedding is $I'_{\text{long}} = \{i'_1, \ldots, i'_{L-2}\}$, and short-term item IDs embedding is $I'_{\text{short}} = \{i'_{L-2-n}, \ldots, i'_{L-2}\}$, where n is the fixed length for the short-term sequence.

After generating new item IDs by clustering, the item frequency $f_{i'}$ is calculated as the number of occurrences of each new item ID, and then inverted to make the model focus on low-frequency items. This inversion is necessary as raw frequency values assign disproportionately high weight to popular items, reinforcing the popularity bias already present in sequential data. By taking the reciprocal of the item count, rare items receive larger values while popular items become smaller, effectively reversing their influence during model training. This transformation provides a simple yet effective way to emphasize underrepresented items without manually tuning reweighting parameters or introducing additional loss terms. The subsequent Min–Max normalization ensures all frequency values fall within a stable numerical range, preventing extreme reciprocal values from dominating the learning process and allowing the frequency signal to integrate smoothly with other embedding features. The item frequency is scaled by a Min-Max scaler,

$$f_{i'} = \frac{\frac{1}{count(i')} - \min(\frac{1}{count(i')})}{\max(\frac{1}{count(i')}) - \min(\frac{1}{count(i')})} \tag{3.4}$$

where $count(i')$ represents the number of times item ID i' appears in the entire collection of item IDs I' in all users' historical interaction sequences.

The item frequency is denoted as $F_{\text{long}} = \{f_1, \ldots, f_{L-2}\}$ for long-term interest, and $F_{\text{short}} = \{f_{L-2-n}, \ldots, f_{L-2}\}$ for short-term interest, where n is the fixed length for the short-term sequence.

3.2.4 Long-term Interest Module

The user's future preferences are influenced not only by recent behaviors but also by long-term interests formed over extended interaction histories. For instance, if a user has consistently purchased athleisure items such as yoga pants, sneakers, and athletic t-shirts throughout the past year, occasional purchases of formal wear (e.g., blazers or shirts) may not significantly alter their underlying preference for athletic apparel. To capture such stable behavioral tendencies, TiIfSRec employs a **dual-gated GRU** in the long-term interest module, which learns the user's general preference patterns and models their temporal evolution over time.

Since the GRU treats sequences as uniformly spaced, a time interval gate T_t is designed as follows to consider the influence of time intervals between user interactions when modeling long-term interest in a GRU. By incorporating both item features and time intervals, this gate allows the model to better reflect how the passage of time affects a user's evolving preferences. The following time gating function T_t integrates the time interval into the GRU to adjust the influence of time interval on the long-term interest of the user. The long-term item embedding reflects the global user preference, including potential interest drift over time, while the time interval captures the temporal context of the current interaction.

$$T_t = \sigma(W_i^T i'_{\text{long } t} + W_{\Delta t}^T \Delta t + b_T) \tag{3.5}$$

where σ function $\sigma(x) = \frac{1}{1+e^{-x}}$ denotes the sigmoid activation function for projection. $i'_{\text{long } t}$ is the long-term item embedding at time step t. Δt is the time interval between the current and the previous interaction in the sequence. W_i^T, $W_{\Delta t}^T$ are learnable projection matrices for long-term interest and time context, and b_T is the bias term for the time gate.

In recommendation scenarios, a small number of popular items dominate user interactions, while many items with potential for personalization appear infrequently and are often ignored by the model. This imbalance in item frequency distribution leads to popularity bias, which reinforces head items while overlooked long-tail items, limiting the model's ability to generate diverse and personalized recommendations. To consider this

imbalance in item frequency, an item frequency gate F_t is designed to improve GRU. The item frequency gate adjusts the effect of the current item on the candidate hidden state based on the frequency of the item. Reduce the hidden state of frequently occurring items to make personalized recommendations instead of hot recommendations. By incorporating the long-term item embedding and item frequency, the model captures the user's overall preferences, reducing the dominance of popular items and enhancing recommendation diversity. The item frequency gate F_t is defined as:

$$F_t = \sigma(W_i^F i'_{\text{long } t} + W_f^F f_t + b_F) \tag{3.6}$$

where σ denotes the sigmoid activation function for projection, f_t is the frequency of the item interacted at time step t. W_i^F, W_f^F are learnable projection matrices for long-term interest and frequency context, and b_F is the bias term for the frequency gate.

While the time gate T_t and the frequency gate F_t provide useful temporal and popularity-related signals, the previous hidden state h_{t-1} in a standard GRU treats all historical information with equal strength. However, long-term user preferences naturally decay over time, and the degree of decay should depend on both the time interval and the frequency characteristics of interacted items. To model this behavior, we introduce a dual-gated adaptive decay mechanism that adjusts the magnitude of the previous hidden state before it is fed into the GRU.

Specifically, the time gate T_t and the frequency gate F_t are first concatenated and projected to an adaptive decay rate:

$$\delta_t = \text{softplus}(W_\delta[T_t \| F_t] + b_\delta) \tag{3.7}$$

where $\delta_t \geq 0$ ensures non-negative decay and W_δ, b_δ are learnable parameters.

The concatenation $[T_t \| F_t]$ acts as a compact summary of how reliable the historical information is at step t. Large time gaps or high item frequencies can push the combined signal toward stronger decay, while short gaps and low-frequency items encourage preservation of past information. The softplus function is used to map this combined signal into a non-negative decay rate, avoiding negative values that would artificially amplify h_{t-1} and ensuring that the model only decides *how much* to forget rather than whether to invert the hidden state.

The decay rate δ_t is then applied to modulate the strength of long-term memory through an exponential attenuation:

$$h_{t-1}^{\text{decay}} = \exp(-\delta_t) \odot h_{t-1} \tag{3.8}$$

where $\odot$ denotes element-wise multiplication. This operation allows long-term preferences to fade at a rate jointly determined by temporal gaps and item-frequency characteristics, providing a more realistic modeling of long-term user interest evolution.

$\exp(-\delta_t)$ serves as a learnable, step-dependent discount factor in $(0, 1]$ applied to each dimension of h_{t-1}. When the time interval is large or the interacted items are very frequent, δ_t becomes larger, $\exp(-\delta_t)$ moves closer to zero, and the corresponding components of h_{t-1} are strongly downweighted. In contrast, for recent interactions on rare items, δ_t remains small, so $\exp(-\delta_t)$ stays near one and the historical signal is largely preserved. In this way, the model avoids using a fixed global decay and instead learns to adapt the strength of long-term memory at each time step based on both temporal and frequency signals.

The reset gate decides the combination of the current input and the previous hidden state. The update gate determines how the current hidden state is updated according to the previous hidden state and the current candidate state. The equations use long-term item embeddings as input:

$$r_t = \sigma_r(W_i^r i'_{\text{long } t} + W_h^r h_{t-1}^{\text{decay}} + b_r) \tag{3.9}$$

$$z_t = \sigma_z(W_i^z i'_{\text{long } t} + W_h^z h_{t-1}^{\text{decay}} + b_z) \tag{3.10}$$

The candidate hidden state $\tilde{h}_t$ represents a potential update to the hidden memory by integrating the current input with the selectively filtered past information. Instead of relying on simple element-wise gating, we introduce a frequency-controlled directional modulation mechanism that enables the model to adjust the direction of the candidate hidden state in the representation space. This design allows less frequent items retain more directional influence, while highly frequent items are prevented from dominating the update. The base candidate hidden state is computed as:

$$h_t^{\text{base}} = \tanh\Big(W_i^h i'_{\text{long } t} + W_h^h(r_t \odot h_{t-1}^{\text{decay}}) + b_h\Big) \tag{3.11}$$

where r_t is the reset gate and h_{t-1}^{decay} is the adaptively decayed previous hidden state.

To incorporate item-frequency effects, a frequency-enhanced candidate representation is computed as:

$$h_t^{\text{freq}} = \tanh\Big(W_{hf}^i i'_{\text{long } t} + W_{hf}^h(r_t \odot h_{t-1}^{\text{decay}}) + W_{hf}^f F_t + b_{hf}\Big) \tag{3.12}$$

where F_t is the frequency gate that encodes the popularity characteristics of the interacted item.

h_t^{freq} serves as a *frequency-aware* alternative to the base candidate state. By feeding the frequency gate F_t through its own set of parameters W_{hf}^i, W_{hf}^h, and W_{hf}^f, the model learns a candidate update whose direction and magnitude are explicitly shaped by item popularity. When the interacted item is rare, F_t can lead h_t^{freq} toward a direction that emphasizes new or underrepresented preference signals; for very frequent items, the same mechanism can dampen overly dominant directions and prevent the hidden state from collapsing toward popular-item patterns. Separating h_t^{base} and h_t^{freq} in this way allows later fusion modules to interpolate between a purely sequence-driven update and a frequency-sensitive update, rather than mixing popularity effects only through simple scaling of a single candidate vector.

A direction gate is then generated to determine how much the frequency-enhanced direction should influence the final candidate hidden state. The candidate hidden state is obtained by softly combining the base and frequency-enhanced representations.

$$\psi_t = \sigma(W_\psi F_t + b_\psi) \tag{3.13}$$

$$\tilde{h}_t = (1 - \psi_t) \odot h_t^{\text{base}} + \psi_t \odot h_t^{\text{freq}} \tag{3.14}$$

where $\odot$ denotes element-wise multiplication. This directional modulation enables the long-term interest module to better capture personalized preference shifts driven by item-frequency patterns.

The direction gate ψ_t therefore acts as a learned soft selector that controls how strongly the frequency-aware signal contributes to the final update. Instead of applying a uniform scaling to the entire hidden state, ψ_t enables *dimension-wise* interpolation between the base update h_t^{base} and the frequency-enhanced update h_t^{freq}. This design allows different latent dimensions to react differently depending on popularity characteristics: some dimensions may emphasize rare-item directions, while others remain aligned with stable sequence-driven patterns.

This soft combination mechanism is crucial because it avoids forcing the model to commit entirely to either the base or the frequency-aware representation. Instead, the GRU update direction can shift gradually as item-frequency patterns evolve, allowing the hidden state to adjust more smoothly to changes in user preference. As a result, the model gains the ability to emphasize useful signals from tail items without destabilizing the learned representation when encountering noisy or highly sparse behaviors.

While GRU effectively models sequential dependencies, its simple update structure lacks a mechanism to adaptively emphasize more informative historical signals. To enhance its ability to filter irrelevant or noisy past interactions, we introduce a relevance-based gating module that reweights the previous hidden state before computing the current update.

At each time step t, the gating coefficient a_t is obtained by projecting the concatenation of the previous hidden state h_{t-1}^{decay} and the current long-term item embedding $i'_{long\ t}$ through a learnable linear layer followed by a sigmoid activation. This gating mechanism performs a local relevance adjustment on h , allowing informative components to be preserved while maintaining less meaningful ones, without performing attention over the entire historical sequence.

$$a_t = \sigma(W_x^a[h_{t-1}^{\text{decay}} \parallel i'_{long\ t}]) \tag{3.15}$$

$$\tilde{h}_{t-1}^{att} = a_t \odot h_{t-1}^{\text{decay}} \tag{3.16}$$

where $\odot$ denotes the element-wise product, $[\cdot \parallel \cdot]$ represents vectors concatenation. W_x^a is a learnable weight matrix for projecting the concatenated features. The σ function denotes the sigmoid activation function for projection.

This relevance gate serves as a lightweight alternative to full sequence attention, enabling the model to highlight useful historical information without incurring the computational cost of attending over all previous states. Since a_t is computed based only on the current input and the decayed hidden state, the model can rapidly determine whether the past information remains relevant at step t. When the interacted item has strong semantic or behavioral similarity to earlier patterns, a_t approaches 1 and retains most of the previous representation. Conversely, if the current item indicates a shift in user intent, a_t is driven toward smaller values, effectively down-weighting outdated components.

By modulating the hidden state in this targeted manner, the gate helps filter noise introduced by infrequent interactions, incorrectly logged events, or behaviors that do not align with the user's evolving interests. This selective preservation of relevant information enhances the quality of long-term preference modeling and ensures that only contextually meaningful features contribute to subsequent GRU updates.

To produce the current hidden state h_t, the attention-enhanced previous hidden state $\tilde{h}_{t-1}^{att}$ is combined with the current candidate hidden state $\tilde{h}_t$ using the update gate z_t. This design allows the model to dynamically balance between preserving important past information captured through attention and incorporating current user interactions.

$$h_t = (1 - z_t) \odot \tilde{h}_{t-1}^{att} + z_t \odot \tilde{h}_t \tag{3.17}$$

where $\odot$ denotes the element-wise product.

The final hidden state, corresponding to the hidden state at the last time step (i.e., a specific instance of h_t), is used to represent the user's long-term interest e_{long}:

$$e_{\text{long}} = h_{\text{final}} \tag{3.18}$$

3.2.5 Short-term Interest Module

Unlike the long-term module, which captures stable user preferences through sequential modeling, the short-term module focuses on recent interactions and dynamic user intents. Therefore, different architectures are adopted to better model their respective characteristics. In many sequence recommendation models, such as Transformer and TiSASRec, researchers use sine and cosine functions to encode absolute position information [17, 40]. However, this fixed encoding method lack flexibility in modeling varying user interaction time intervals. Compared to the fixed time encoding method of sine and cosine functions, GRU can capture more complex temporal relationships more flexibly due to its recursive structure when encoding time interval information. Gated Recurrent Unit (GRU) is a type of recurrent neural network designed to model sequential data by maintaining and updating a hidden state over time [9]. At each time step, the GRU takes the current time interval as input, combines it with the previous hidden state, and updates the current hidden state through nonlinear transformations. Since time intervals are ordered by sequence, the GRU could learn the position sequence information so that it does not need to encode separately. To address the limitations of lack of flexibility and the need for additional positional encoding, GRU is adopted to encode the time interval information in a learnable and dynamic way.

The update process of the GRU cell when encoding the time interval sequence is defined as follows. The hidden state h_t is a dynamic representation of all relevant time interval information in the sequence up to time step t. It is updated at each step to incorporate new input while retaining useful historical context. The reset gate r controls how much of the previous hidden state should be forgotten, and the update gate z determines how much of the previous hidden state should be preserved. The candidate hidden state h'_t proposes a potential update to the memory by combining the current input and selective past information.

$$r_t = \sigma(W_x^r \Delta t'_t + W_h^r h_{t-1} + b_r) \tag{3.19}$$
$$z_t = \sigma(W_x^z \Delta t'_t + W_h^z h_{t-1} + b_z) \tag{3.20}$$
$$h'_t = \tanh(W_x^c \Delta t'_t + W_h^c (r_t \odot h_{t-1}) + b_c) \tag{3.21}$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{3.22}$$

where $\odot$ denotes the element-wise product between two vectors. r_t, z_t, h'_t denote the reset gate, update gate, and candidate state of the current interaction at time step t. W_x^r, W_x^z, W_x^c are weight matrices of the input $\Delta t'_t$ for reset gate, update gate, and candidate state. W_h^r, W_h^z, W_h^c are weight matrices of the previous hidden state h_{t-1} for reset gate,

update gate, and candidate state. b_r, b_z, b_c are bias terms for reset gate, update gate, and candidate hidden state. Sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$ and tanh activation function $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ are used for projection. $\tanh(x)$ is used for the candidate hidden state because it can represent both positive and negative values, which provides richer expressive power.

By the final time step, the GRU outputs a hidden state h_n, which encodes the overall temporal information across the entire short-term sequence.

To further enhance the short-term interest representation, the time interval and item frequency information are then integrated into a self-attention mechanism, which allows the model to focus on both the time-based and frequency-based aspects of user interactions.

The attention mechanism is defined with three key components: the query Q, key K, and value V. The query Q represents the current focus, the key K is a reference for matching, and the value V contains the item representations including the ID features, temporal signals, and frequency information. The attention weights are computed based on the similarity between Q and K. Then attention weights are used to determine how much attention to pay to each corresponding value. This enables the model to selectively combine information that is most relevant to the current context [35].

The query Q is mainly used to determine the user's interest at the current step, so it does not consider the time interval and item frequency information, thus avoiding making the query information more complex and interfering with the model's effective focus on the current moment. While to incorporate temporal and frequency information, the key K and value V are further updated with time interval and item frequency embeddings as follows:

$$Q = W_Q I'_{\text{short}} \tag{3.23}$$

$$K = W_K I'_{\text{short}} + h_n + F_{\text{short}} \tag{3.24}$$

$$V = W_V I'_{\text{short}} + h_n + F_{\text{short}} \tag{3.25}$$

where h_n denotes the temporal embedding generated by the GRU's last time step, and F_{short} represents the short-term item frequency embedding. I'_{short} is the embedding of short-term item IDs. W_Q, W_K, W_V are learnable weight matrices used to transform the input sequence into query, key, and value representations.

To capture the user's dynamic interest, the self-attention weight A is calculated to aggregate information from the short-term interaction sequence. The self-attention weight determines how much attention each item should receive relative to the current query by

calculating the similarity between query Q and key K [35]:

$$A = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) \tag{3.26}$$

where d denotes the dimension of the query and key vectors. $\sqrt{d}$ is used to avoid large values of the inner product and keep the gradient value stable. $\top$ is the transpose of the key matrix. The softmax function $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$ is used to normalize the attention scores into a probability distribution.

The short-term interest representation e_{short} is obtained by multiplying the attention weight A with the value V to distinguish more relevant interactions from less significant ones. This operation assigns higher weights to the value vectors that are more relevant to the user's current interest, allowing the model to focus on the most important interactions within the short-term sequence. Therefore, the model provides a dynamic and personalized representation of the user's recent preferences.

$$e_{\text{short}} = AV \tag{3.27}$$

3.2.6 Fusion Gate

In recommender systems, users' interests are usually multi-dimensional and varying, including long-term and short-term interests. Long-term interests reflect users' stable preference and behavioral patterns, while short-term interests reflect users' current interests. In some cases, users' long-term interests, such as their preference for a particular brand, may become dominant, and the weight of long-term interests need to be increased. In other cases, a user's short-term interests may change rapidly, such as the current seasonal trend, and the short-term interests may need to be weighted more heavily.

To capture user interests more accurately, a gate mechanism is used to adaptively blend short-term and long-term interests. In training stage, the first $|i| - 2$ sequence of user interactions $\{i_1, ..., i_{|i|-2}\}$ is to predict the item that the $|i| - 1$ user is likely to be interested in. The gating mechanism enables the model to dynamically adjust the relative importance of long-term and short-term interests based on the current context, improving the accuracy of user interest prediction. It computes a weight vector g to control the relative influence of long-term and short-term interest to the final user representation, enabling model focus on the most relevant signals in different scenarios.

$$g = \sigma(W_l^g e_{\text{long}} + W_s^g e_{\text{short}} + b_g) \tag{3.28}$$

where e_{long} represents long-term interest embedding generated by Equation 3.18, and e_{short} represents short-term interest embedding produced by Equation 3.27. W_l^g and W_s^g denote the learnable weight matrices, and b_g is the bias term for weight vector g.

The final interest representation y is obtained by adaptively weighting the short-term and long-term embeddings based on the gating weight vector g, which dynamically determines the relative contribution of each interest type. This design allows the model to emphasize short-term behaviors when recent interactions are more informative, and to focus on long-term preferences when the user's historical patterns are more predictive.

$$y = g \odot e_{\text{long}} + (1 - g) \odot e_{\text{short}} \tag{3.29}$$

where y denotes the predicted items.

The model is trained to predict the next interacted item via a softmax layer over all candidate items. The objective function is a cross-entropy loss computed as:

$$\mathcal{L} = -\sum_{(u,i_t)} \log P(i_t \mid y_u) \tag{3.30}$$

where y_u is the fused user representation in Equation 3.29, and $P(i_t)$ denotes the probability of item i_t being the next interaction.

3.3 Experimental Evaluation

This section reports the experimental evaluation of TiIfSRec. The analysis includes comparisons with strong baseline models, performance measurements across different application domains, and ablation studies to verify the contribution of each major component. These results help clarify how and why TiIfSRec improves sequential recommendation performance. This chapter evaluates TiIfSRec on the Amazon Reviews 2023 dataset [10] across three representative domains: `Cell_Phones_and_Accessories`, `Movies_and_TV`, and `Clothing_Shoes_and_Jewelry`. These domains exhibit distinct behavioral patterns and category structures, allowing a comprehensive assessment of long–short interest modeling, temporal sensitivity, and frequency-awareness. The model is benchmarked against four widely adopted sequential recommendation baselines—GRU4Rec, MARank, SASRec, TiSASRec—to examine how well TiIfSRec improves next-item prediction relative to RNN-based, attention-based, Markov-based, and gating-based architectures.

Evaluation is conducted using four standard top-k ranking metrics (Precision@10, Recall@10, NDCG@10, and MAP@10), which jointly quantify retrieval accuracy, ranking

quality, and position-sensitive relevance. By comparing TiIfSRec with these diverse baselines under consistent datasets, domains, and metrics, the experiments aim to validate the effectiveness of integrating time-interval modeling, frequency-aware gating, and dual-stage interest fusion within a unified sequential recommendation framework.

To analyze the effect of representation capacity, the embedding dimension **K** is varied, which controls the size of the latent vector used to encode user and item features. Higher dimensions can improve performance but may also increase overfitting and computation cost. All hyper-parameters are selected within reasonable ranges and tuned based on validation performance. Key parameters such as embedding dimension, learning rate, and batch size are adjusted to achieve a balance between model capacity and generalization.

To further evaluate the impact of different architectural components and design choices, an ablation study is conducted in the following section. This analysis systematically examines the effect of removing or modifying key modules, as illustrated in Figure 3.3, to validate the contribution of each part to the overall performance.

3.3.1 Dataset

The experiment is performed based on the Amazon Reviews 2023 dataset, a large-scale collection that includes 571.54 million reviews spanning from May 1996 to September 2023. The dataset contains rich user interaction histories with detailed timestamps, diverse product metadata, and cross-domain item distributions. Compared with previous Amazon datasets, the 2023 version provides substantially expanded item coverage, cleaner metadata processing, and finer-grained temporal information recorded at the second level. These improvements allow more accurate modeling of user behavior sequences, especially in scenarios where the recency of interactions and the temporal spacing between actions are critical for interest prediction.

Compared with earlier public Amazon datasets from 2013, 2014, and 2018, the 2023 release also introduces higher-quality product annotations and more consistent formatting across domains. The improvement in metadata completeness—such as standardized titles, consolidated product families, and reduced noise in category labels—offers a more reliable basis for extracting textual features. This consistency enables TF-IDF–based clustering to produce more meaningful semantic groupings and reduces ambiguity in item identity. In addition, the considerable increase in the number of unique products strengthens the representation of long-tail items, providing a more realistic environment for evaluating models that aim to address frequency imbalance. The inclusion of second-level timestamps further supports time-interval modeling by capturing subtle behavioral shifts that older

versions could not reflect. Overall, these characteristics make the 2023 dataset particularly suitable for studying scenarios where temporal sensitivity, semantic clustering quality, and long-tail distribution patterns jointly influence recommendation performance.

The three selected domains exhibit distinct behavioral characteristics and interaction dynamics. These domain-specific characteristics are derived from empirical analysis of the dataset and are consistent with commonly observed patterns in recommender systems, such as long-tail distributions and temporal dynamics in user behavior [17, 21].

Cell_Phones_and_Accessories contains 11.6 million users and 1.3 million items, reflecting a technologically oriented market with rapid product updates and strong sensitivity to short-term interests. User behavior in this category often includes bursts of activity around device upgrades or accessory purchases. Items frequently become obsolete as new models are released, making temporal recency and short-term intent cues particularly important for prediction. The domain also shows moderate long-tail effects, where niche accessories appear infrequently but may have strong contextual relevance.

Movies_and_TV consists of 6.5 million users and 747.8 thousand items, representing a consumption pattern that is less tied to product lifecycle and more influenced by stable genre preferences. Users typically engage in continuous and repetitive consumption of similar types of media, causing long-term preference signals to be highly predictive. Compared with other domains, the item catalog turns over slowly, and popularity distributions are heavily skewed, with a small number of blockbusters dominating interactions. This domain therefore provides a strong scenario for evaluating long-term modeling and popularity-awareness.

Clothing_Shoes_and_Jewelry includes 22.6 million users and 7.2 million items, making it one of the largest and most diverse product categories. User purchase patterns exhibit both seasonal fluctuations and strong long-tail effects due to the wide variety of styles, brands, and sizes. Items frequently appear with low interaction counts, creating severe sparsity. User preferences can shift quickly due to fashion trends or temporal factors such as holidays, making this domain challenging for both frequency-aware modeling and short-term intent tracking. Its large item space also makes clustering-based ID generation particularly beneficial.

In this chapter, four fields are selected as they provide the essential information for sequential modeling, including user identity, temporal order, interaction validity, and product grouping, while avoiding irrelevant or noisy attributes. The date fields include:

- `user_id`: ID of the reviewer.
- `timestamp`: Time of the review (unix time).

- `verified_purchase`: User purchase verification.
- `parent_asin`: Parent ID of the product. It groups product variants such as colors, sizes, or styles under a common product family.

3.3.2 Evaluation Metrics

TiIfSRec is evaluated using four widely adopted top-k recommendation metrics. These evaluation measures quantify different aspects of ranking quality and allow consistent comparison across all baseline models used in the experiments.

- **Precision@10**: It calculates how many of the top 10 recommendations are actual interested items for the user, and takes the average over all N users.

$$\text{Precision@10} = \frac{1}{N}\sum_{u=1}^{N}\frac{\textit{Number of relevant items in top10}}{10} \tag{3.31}$$

- **Recall@10**: It measures the proportion of relevant items that a recommender system can find out of the top 10 items that user interested in. It takes the average over all N users.

$$\text{Recall@10} = \frac{1}{N}\sum_{u=1}^{N}\frac{\textit{Number of relevant items in top10}}{\textit{Total number of relevant items}} \tag{3.32}$$

- **NDCG@10**: NDCG is a position-aware metric which assigns larger weights on higher positions [41]. It evaluates the ranking quality of relevant items in the recommendation list.

$$\text{NDCG@10} = \frac{\text{DCG@10}}{\text{IDCG@10}} = \frac{\sum_{i=1}^{10}\frac{rel(i)}{\log_2(i+1)}}{\sum_{i=1}^{10}\frac{rel^*(i)}{\log_2(i+1)}} \tag{3.33}$$

  where $rel(i)$ is the relevance score for the i-th position, and $rel^*(i)$ is the relevance score of the i-th position in the ideal case.

- **MAP@10**: MAP measures the overall ranking quality by averaging the precision across all relevant items. It provides a more global evaluation of the recommendation list, rather than focusing only on specific positions, making it more informative for assessing ranking performance.

$$\text{MAP@10} = \frac{1}{N}\sum_{u=1}^{N} AP@10(u) \tag{3.34}$$

where $AP@10(u)$ represents the Average Precision of user u up to the top 10 recommended items.

These four metrics complement one another and collectively provide a comprehensive evaluation of TiIfSRec. Precision@10 and Recall@10 quantify the accuracy and coverage of retrieved items, revealing how effectively the model identifies relevant results within a fixed recommendation list. NDCG@10 incorporates position-aware weighting, highlighting whether the model ranks the most relevant items near the top. MAP@10 further emphasizes the ordering of relevant items across the top-10 list by averaging precision across hit positions. Overall, these metrics capture both retrieval quality and ranking consistency, ensuring a robust and multi-perspective assessment of top-k recommendation performance.

3.3.3 Baselines

In this chapter, TiIfSRec is compared against 4 representative baselines. As TiIfSRec is proposed in the context of sequential data, these methods are all sequential models. Here are brief descriptions of these models:

- **GRU4Rec** [9]: This is a session-based sequential recommendation baseline. It uses GRU network to predict the next item that the user may be interested in.
- **MARank** [43]: This is a sequential recommendation baseline with multi-order attention mechanism. It fine-grained captures short-term interests from both the union and individual level.
- **SASRec** [14]: This is a self-attention-based baseline for sequential recommendation. It uses self-attention mechanism. At each time step, it explores which items are "relevant" from the user's behavior history and uses them to predict the next item.
- **TiSASRec** [17]: This is a state-of-art sequential recommendation baseline that was proposed recently. It first combines the time interval information into self-attention mechanism. At each time step, it explores which items are "relevant" from the user's behavior history and uses them to predict the next item.

These baselines are selected to represent different types of sequential recommendation models. They cover a diverse range of sequential modeling paradigms, including recurrent networks, multi-order attention, self-attention, and time-aware attention mechanisms. Comparing TiIfSRec against these representative methods allows us to evaluate its advantages in modeling temporal dynamics, capturing multi-scale interests, and handling item-frequency imbalance under realistic sequential recommendation settings.

3.3.4 Results and Analysis

In this section, we closely examine the empirical behavior of TiIfSRec and compare its performance with representative sequential recommendation baselines across three domains. The goal is to understand not only whether TiIfSRec improves top-k recommendation accuracy, but also why these improvements occur under different behavioral patterns, sparsity levels, and temporal characteristics. By analyzing metric trends, domain-specific differences, and the relative performance gaps between models, we can better interpret the contributions of each modeling component and identify the scenarios where time–frequency co-modeling provides the most substantial benefits.

Table 3.1: Result of experiment on TiIfSRec

Dataset	**Method**	**Prec@10**	**Recall@10**	**NDCG@10**	**MAP@10**
Cellphone	GRU4Rec	0.01666667	0.16666667	0.06706044	0.03893519
	MARank	0.01833333	0.18333334	0.06766283	0.03406085
	SASRec	0.02166667	0.21666667	0.11233173	0.08013889
	TiSASRec	0.02333333	0.23333333	0.11904398	0.08591931
	TiIfSRec	**0.02500000**	**0.25000000**	**0.12595006**	**0.08955688**
	Improvement	**7.14%**	**7.14%**	**5.80%**	**4.23%**
Clothes	GRU4Rec	0.01500000	0.15000001	0.06804362	0.04347222
	MARank	0.01500000	0.15000001	0.08393995	0.06537037
	SASRec	0.01833333	0.18333334	0.09664425	0.06972222
	TiSASRec	0.02000000	0.20000000	0.11268687	0.08488095
	TiIfSRec	**0.02166667**	**0.21666667**	**0.12277264**	**0.09335317**
	Improvement	**8.33%**	**8.33%**	**8.95%**	**10.00%**
Movies	GRU4Rec	0.05500000	0.55000001	0.47834228	0.45486111
	MARank	0.05833333	0.58333331	0.52380877	0.50615079
	SASRec	0.06000000	0.60000002	0.53245737	0.51138889
	TiSASRec	0.06166666	0.61666667	0.53870410	0.51497354
	TiIfSRec	**0.06500001**	**0.65000000**	**0.55410093**	**0.52337302**
	Improvement	**5.41%**	**5.41%**	**2.86%**	**1.63%**

Table 3.1 summarizes the experimental results based on three datasets. As shown in Table 3.1, TiIfSRec achieves the best performance across all three datasets on Precision@10, Recall@10, NDCG@10, and MAP@10. Prec@10 denotes Precision@10 defined in Equation 3.31. Marked values indicate the best baseline used for TiIfSRec improvement calculation. The overall improvements show that modeling both time intervals and

item-frequency imbalance helps the model better understand user preferences over long sequences.

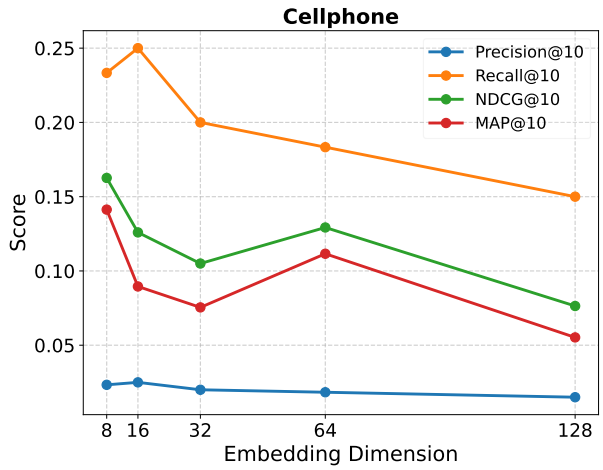
Across different datasets, the performance gap between models reflects the characteristics of each domain. For the Movies dataset, user behavior is relatively dense and stable—people tend to watch many movies within similar genres—so all models achieve fairly high scores. Even in this easier setting, TiIfSRec still makes noticeable improvements, which suggests that frequency-aware long-term modeling can capture additional patterns that traditional sequence models may miss. The Clothes dataset is much more challenging due to high sparsity and strong long-tail effects. Users rarely interact repeatedly with similar items, and their preferences change frequently. Therefore, all models perform lower overall. However, TiIfSRec shows the largest improvement here. By explicitly handling item-frequency imbalance, the model avoids being overly biased toward popular items, which helps it make more accurate predictions for users whose histories contain more infrequent or diverse items. The Cellphone dataset lies somewhere between the other two. User preferences are more stable than in fashion (e.g., staying within the same device ecosystem), but the data is still not as dense as Movies. TiIfSRec again outperforms all baselines, indicating that both the temporal and frequency components are useful in this moderately sparse scenario.

Comparing TiIfSRec directly with TiSASRec which is the strongest baseline that also models time intervals, TiIfSRec consistently performs better on all metrics and datasets. This shows that time information alone is not enough. Frequency imbalance in long sequences also plays a key role. By reducing the dominance of highly frequent items and giving more balanced attention to long-tail items, TiIfSRec produces more accurate and diverse recommendations.

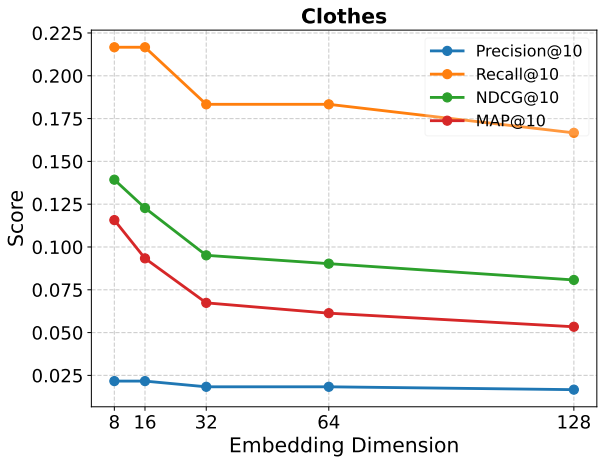
Overall, although the absolute metric values are affected by dataset sparsity, the consistent improvements across all datasets demonstrate that combining time intervals with frequency-aware long-term modeling helps the system form more reliable user representations and generate better top-k recommendations.

Figure 3.2 reports the performance of TiIfSRec under different embedding dimensions K on the three datasets. Overall, the results show a consistent pattern: smaller embedding sizes (8 or 16) lead to the best performance, while large dimensions (64 and 128) tend to reduce the performance.

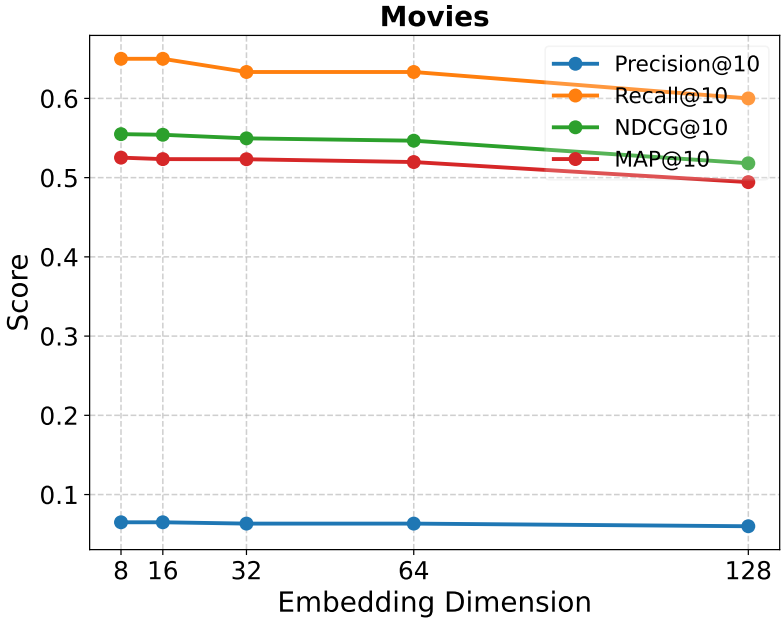
For Cellphone, all four metrics peak at $K = 16$, showing that a compact representation is sufficient to capture the relatively simple behavioral patterns in this small dataset. Larger embeddings introduce noise and overfitting, causing both NDCG@10 and MAP@10 to decline sharply. For Clothes, the trend is similar: $K = 8, 16$ produces the best perfor-



(a) Effect of embedding dimension on TiIfSRec in CellPhone



(b) Effect of embedding dimension on TiIfSRec in Clothes



(c) Effect of embedding dimension on TiIfSRec Movies

Figure 3.2: Effect of the embedding dimension K on TiIfSRec across three domains

mance, while dimensions above 32 cause a gradual degradation across all metrics. This suggests that user interactions in this dataset are not complex enough to benefit from high-dimensional embeddings, and larger K instead harms generalization. For Movies, although the absolute scores are much higher due to richer and denser interactions, the same pattern still appears. Performance is stable at $K = 8, 16, 32$ but begins to drop when K increases to 64 and 128. The consistent decline in NDCG@10 and MAP@10 indicates that excessively large embeddings lead to redundant parameters that do not provide additional useful information.

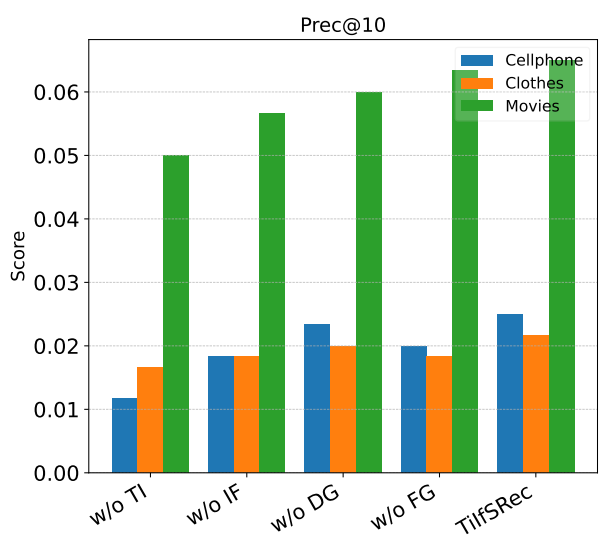
Across all datasets, smaller embeddings not only achieve the best results but also reduce model complexity and training cost. This confirms that for datasets of this scale, compact embeddings are more effective than larger ones, and setting K between 8 and 16 provides the best balance between expressiveness and generalization.

Table 3.2: Relative improvement of TiIfSRec over the w/o DG variant on three domains

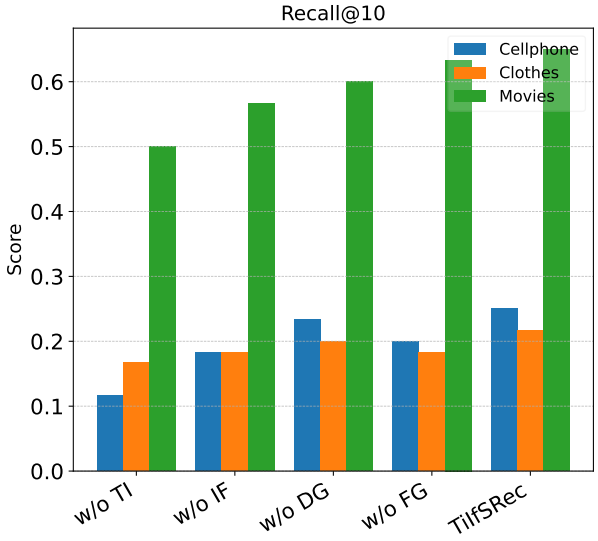
Metric	**Cell Phones**	**Clothing**	**Movies**
Precision@10	7.30%	8.33%	8.33%
Recall@10	7.14%	8.33%	8.33%
NDCG@10	1.95%	10.10%	2.26%
MAP@10	0.55%	10.70%	0.22%

Furthermore, to verify the contribution of each component in TiIfSRec, we conducted an ablation study by successively removing or modifying key modules. Specifically, we compared four model variants: without time-interval modeling (w/o TI), without item-frequency modeling (w/o IF), without dual gating in the long-term module (w/o DG), and without the fusion gate (w/o FG). As shown in Figure 3.3, all four variants exhibit reduction compared with the full model, and the performance gaps reveal the importance of each component.

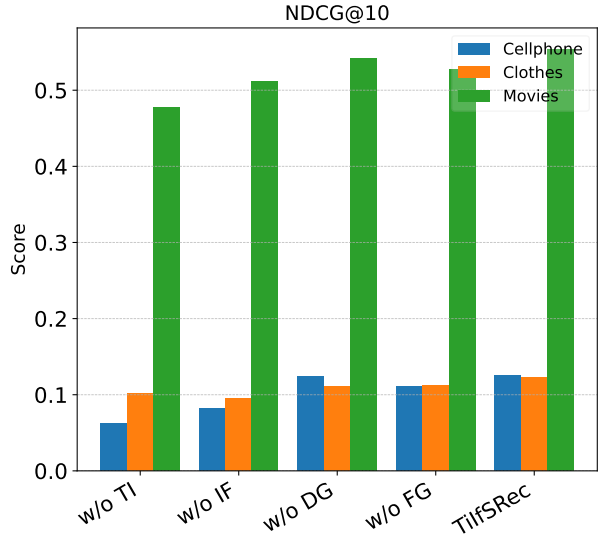
Removing time-interval modeling (w/o TI) leads to the most significant drop across all datasets, especially on Movies, where temporal patterns are stronger. This confirms that explicitly modeling the spacing between interactions is essential for capturing long-term preference evolution rather than treating all historical behaviors as equally important. Eliminating item-frequency modeling (w/o IF) also reduces performance on every metric. Without this module, the model cannot correct the inherent frequency imbalance in real datasets, resulting in over-reliance on popular items and weaker personalization, particularly on Clothes and Cellphone where the long-tail issue is more pronounced. Removing



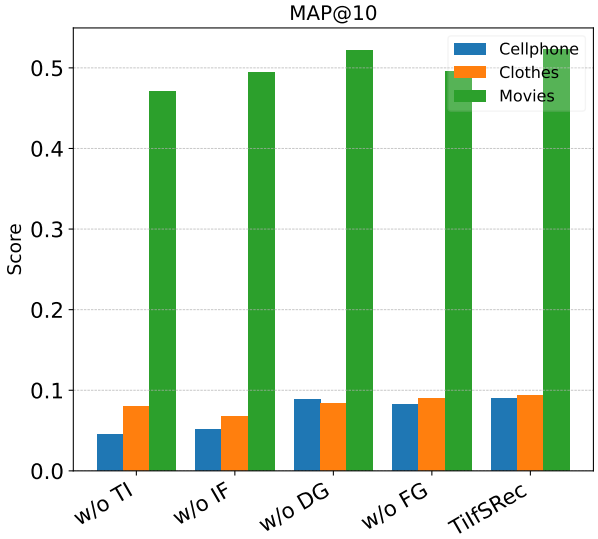
(a) Precision@10 comparison in TiIfSRec ablation variants



(b) Recall@10 comparison in TiIfSRec ablation variants



(c) NDCG@10 comparison in TiIfSRec ablation variants



(d) MAP@10 comparison in TiIfSRec ablation variants

Figure 3.3: Ablation study of TiIfSRec and its variants

dual gating in the long-term module (w/o DG) reduces the performance. Since the dual-gated design jointly controls temporal decay and frequency-driven directional adjustment, removing it prevents the model from properly filtering and shaping long-term signals, causing both NDCG and MAP to drop. Finally, removing the fusion gate (w/o FG) produces slightly worse results than the full model but remains stronger than the other variants. Although simple addition of long- and short-term features still works reasonably well, the absence of an adaptive fusion mechanism limits the model's ability to balance stable preferences and recent behaviors.

In Table 3.2, TiIfSRec is compared with the w/o DG variant to isolate the contribution of the dual-gating mechanism. The results show that TiIfSRec consistently outperforms w/o DG across all metrics and domains, though the magnitude of improvement varies depending on dataset characteristics. On the Cell Phones and Clothing datasets, which exhibit higher item sparsity and stronger long-tail distributions, the gains in Precision and Recall exceed 7–8%, suggesting that dual gating effectively enhances the stability of long-term preference modeling in sparse environments. In contrast, Movies shows a smaller but still steady improvement, with around 8% gains in Precision and Recall but relatively modest increases in NDCG and MAP. This reflects the denser interaction patterns in the Movies domain, where temporal order plays a stronger role and high-frequency items dominate, making directional filtering less drastic but still beneficial. The NDCG and MAP improvements further highlight that dual gating contributes not only to identifying the correct next item but also to ranking it in a more accurate position. In particular, Clothing shows a significant boost in both NDCG (+10.10%) and MAP (+10.70%), demonstrating that frequency-sensitive directional adjustments have the strongest impact in domains where popularity imbalance is severe. The table confirms that the dual-gating mechanism contributes meaningfully across all evaluation dimensions—accuracy, recall, ranking quality, and top-k relevance—reinforcing its necessity within the long-term interest encoder of TiIfSRec.

Overall, the consistent improvements of TiIfSRec across all metrics validate the effectiveness of its core components. Each module—time intervals, item frequency, dual gating, and fusion—contributes meaningfully, and together they enable the model to better capture long-term user interests and generate more accurate recommendations.

3.4 Summary

In this chapter, we presented TiIfSRec, a Dual-Gated Time–Frequency Co-Modeling framework for sequential recommendation. The model jointly incorporates time intervals, item-

frequency imbalance, and multi-scale user interests to better capture the dynamics of real-world behavioral sequences. To alleviate the sparsity of raw item IDs, TF-IDF–based feature extraction and KMeans clustering were used to generate condensed item categories, enabling more robust learning in domains with heavy long-tail distributions.

For long-term interest modeling, TiIfSRec enhances a GRU backbone with a time gate and a frequency gate that jointly control temporal decay and frequency-aware preference adjustment. A relevance-based attention module further highlights informative historical behaviors and suppresses noisy or obsolete signals. For short-term interest modeling, a time-interval–aware GRU is designed to encode recency and positional shifts, and its output is refined through a self-attention layer to emphasize immediate user intent. By integrating these components, TiIfSRec maintains an adaptive balance between stable preference tendencies and rapidly evolving short-term motivations.

Experimental results on three representative Amazon domains demonstrate that TiIfSRec consistently outperforms several strong sequential recommendation baselines across four top-k evaluation metrics. The ablation studies confirm the contribution of each architectural module, showing that long–short interest fusion benefits significantly from explicit temporal modeling, frequency-aware gating, and relevance-based filtering mechanisms.

Although TiIfSRec effectively improves long- and short-term interest modeling, its fusion strategy still relies on deterministic gating and single-step combination. This limitation restricts the model's ability to represent multi-modal or uncertain user intentions, especially in situations where short-term behaviors fluctuate rapidly or when conflicting preference signals coexist. To address these challenges, the next chapter introduces DiffLSRec, a diffusion-based generative fusion framework designed to iteratively refine user interest representations. By viewing the fusion process as a progressive denoising trajectory rather than a static combination, DiffLSRec aims to model multiple plausible future preferences and capture richer long–short interactions beyond deterministic gating mechanisms.

Chapter 4

Diffusion-based Long and Short Term Interest Sequence Recommendation

4.1 Introduction

In this chapter, we introduce DiffLSRec, a diffusion-based framework designed to achieve adaptive fusion of long-term preferences and short-term intentions in sequential recommendation. Unlike conventional models that integrate these two signals through fixed weighting rules, static gates, or one-shot aggregation, DiffLSRec reformulates interest fusion as a gradual generative refinement process. By leveraging a multi-step denoising trajectory, the model can continuously adjust the relative importance of stable historical tendencies and recent behavioral patterns based on temporal uncertainty and contextual variation.

Existing sequential recommenders typically rely on recurrent units or attention mechanisms to encode behavior sequences. While these architectures are capable of learning dependency structures, their fusion strategies are typically static and rely on a single-step combination of long-term and short-term signals. They treat long-term and short-term signals as static features and combine them in a single computation step, often leading to limited adaptability when user behavior shifts abruptly. More importantly, aggregated short-term embeddings frequently overlook token-level nuances—details that may be crucial for capturing transient intents or subtle behavioral cues. These limitations are amplified in domains where user interests evolve rapidly, timestamps are unevenly distributed, or item preferences exhibit strong long-tail characteristics.

To address these shortcomings, DiffLSRec places the fusion process within a diffusion-based generative modeling framework. The model begins with the long-term representation

as a prior initialization and iteratively refines it through a reverse diffusion process guided by short-term intent. Instead of enforcing a fixed interaction between the two signals, the refinement evolves step-by-step, allowing the model to respond dynamically to uncertainty at different stages of denoising. Early steps emphasize robust long-term preferences, while later steps increasingly incorporate short-term contextual clues as noise diminishes, producing a more fine-grained and intent-aware representation.

To enhance the expressiveness and controllability of this generative refinement, DiffLSRec introduces two core components. Token-level Contextual Enhancement applies cross-attention to the most recent interaction tokens, enriching the short-term guidance signal with structure that would otherwise be lost in pooled embeddings. This allows the model to capture fine-grained behavioral dynamics without overwhelming the long-term representation. In addition, Monotonic SNR-Adaptive Guidance modulates the fusion strength according to the signal-to-noise ratio at each diffusion step, ensuring a smooth and interpretable transition from coarse long-term priors to precise short-term adaptations. Together, these mechanisms give DiffLSRec flexibility to model gradual preference changes and robustness to noise, sparsity, and temporal irregularity.

The key contributions of this chapter are summarized as follows:

- It propose a diffusion-based sequential recommendation framework that formulates long–short interest fusion as a multi-step generative refinement process rather than a single-step deterministic operation.
- It design a progressive denoising mechanism where the long-term embedding acts as a generative starting point and short-term intent provides conditional guidance at each stage, enabling dynamic rebalancing of temporal signals throughout the refinement trajectory.
- It introduce two complementary modules—Token-level Contextual Enhancement for capturing nuanced recent behavioral cues, and Monotonic SNR-Adaptive Guidance for controlling the contribution of long- and short-term components based on uncertainty levels in the diffusion process.

The remainder of this chapter is organized as follows. Section 4.2 describes the theoretical formulation and architectural components of DiffLSRec, including the forward diffusion, reverse denoising, contextual enhancement, and SNR-guided fusion. Section 4.3 outlines the experimental design, datasets, baselines, and evaluation settings. It reports and analyzes empirical results across multiple domains. Section 4.4 discusses limitations,

insights, and practical considerations for applying diffusion models in recommendation. It concludes the chapter and connects the proposed framework to future generative modeling extensions.

4.2 Methodology

4.2.1 Architecture

The proposed DiffLSRec model captures and integrates both long-term and short-term user interests through a diffusion-based generative framework. Rather than relying on static fusion strategies such as weighted sums, gating, or attention, DiffLSRec employs a multi-step denoising process in which short-term signals progressively guide the refinement of long-term representations. To further enhance contextual understanding and controllability, DiffLSRec incorporates two mechanisms: Token-level Contextual Enhancement, which attends to fine-grained sequential tokens via cross-attention, and Monotonic SNR-Adaptive Guidance, which regulates the balance between long- and short-term influences according to the signal-to-noise ratio during diffusion.

The overall architecture of DiffLSRec is illustrated in Figure 4.1. The embedding layer first transforms each user's interaction sequence into dense vectors. From this sequence, two levels of user interests are extracted: long-term interests from the complete interaction history and short-term interests from the most recent behaviors. These embeddings are then passed to the diffusion-based fusion module, where the long-term representation acts as a generative prior undergoing forward diffusion, while the short-term representation provides conditional guidance during reverse denoising. This design allows the model to iteratively balance stable historical preferences with rapidly changing short-term intents. The final denoised representation is subsequently used to compute personalized recommendation scores for candidate items.

4.2.2 Embedding

To make the user interaction data accessible for modeling, DiffLSRec embeds the raw inputs such as user, item, and time information into a low-dimensional latent space while preserving their latent patterns and correlations.

Let U be the set of users and I the set of items. Each interaction is represented as a tuple (u, i, t), where $u \in U$ is the user identifier, $i \in I$ is the interacted item, and t is the

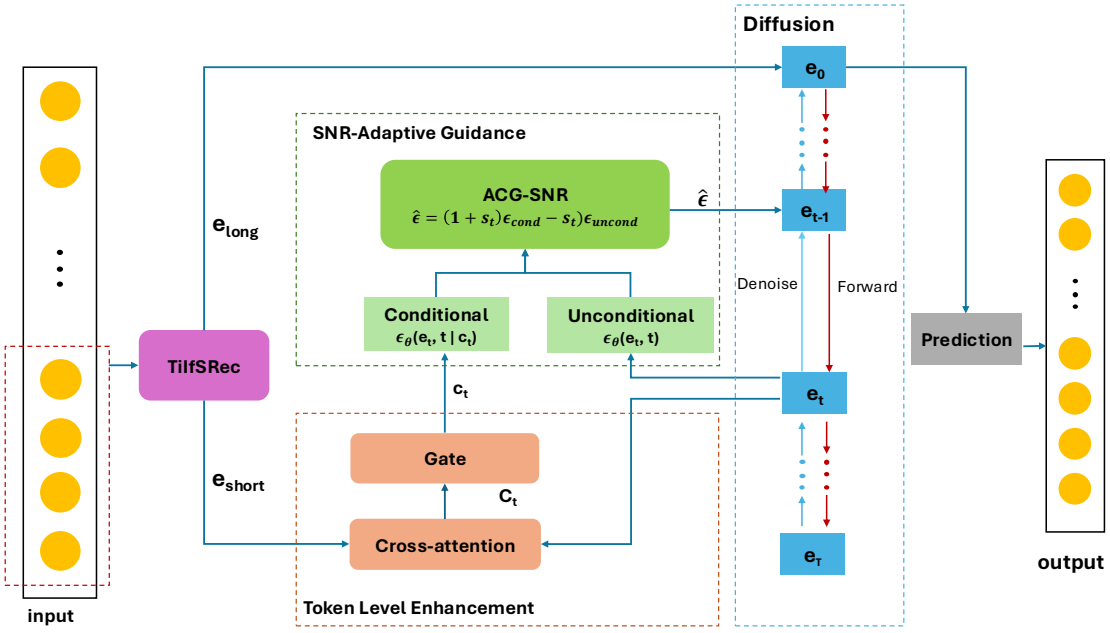


Figure 4.1: The overview framework of DiffLSRec model.

interaction timestamp. For each user $u \in U$, the interaction history can be formulated as a sequence of items ordered by time, denoted by $\{i_1, i_2, \ldots, i_L\}$, with the corresponding timestamp sequence $\{t_1, t_2, \ldots, t_L\}$.

For each item $i \in I$, an embedding vector $e_i \in \mathbb{R}^d$ is assigned and stored in a shared embedding matrix $E \in \mathbb{R}^{|I| \times d}$. To capture temporal patterns, the sequence of timestamps is transformed into time intervals, defined as

$$\Delta t_j = |t_j - t_{j-1}|, \quad j \geq 2,$$

where the first interval is set to $\Delta t_1 = \Delta t_2$ to avoid undefined values. To ensure stable training, time intervals are scaled into the range $[0, 1]$ using a Min–Max normalization, following the same normalization strategy described in Section 3.2.2:

$$\Delta t'_j = \frac{\Delta t_j - \Delta t_{\min}}{\Delta t_{\max} - \Delta t_{\min}} \tag{4.1}$$

In addition, item frequency is considered to reflect how often each item appears in the training corpus. The frequency value f_i for item i is normalized into $[0, 1]$ and mapped into a trainable frequency embedding vector $e_{f_i} \in \mathbb{R}^d$. This provides complementary information beyond the item ID and timestamp.

To build user representations, the interaction history is devided into two parts. For long-term sequence, $I_{\text{long}} = \{i_1, \ldots, i_{L-2}\}$ with corresponding temporal and frequency features, capturing stable user preferences. For Short-term sequence, $I_{\text{short}} = \{i_{L-2-n}, \ldots, i_{L-2}\}$ focusing on the most recent n interactions. If the sequence is shorter than n, left padding is applied; otherwise, only the last n items are preserved.

The corresponding time intervals and frequency sequences are partitioned in the same way. The two user representations are then obtained as

$$e_{long} = F_{long}(I_{\text{long}}, \Delta t'_{\text{long}}, f_{\text{long}}) \tag{4.2}$$

$$e_{short} = F_{short}(I_{\text{short}}, \Delta t'_{\text{short}}, f_{\text{short}}) \tag{4.3}$$

where $F_{long}(\cdot)$ denotes the long-term encoder, implemented with a GRU-based architecture enhanced by time- and frequency-aware gates. This design allows the hidden states to capture sequential dependencies while adaptively adjusting for varying temporal intervals and item popularity. Similarly, $F_{short}(\cdot)$ is the short-term encoder, which applies a self-attention mechanism over the most recent items, combined with temporal and frequency embeddings. This enables the model to highlight the most relevant behaviors within the current context. The detailed formulations of the long-term interest encoder $F_{long}(\cdot)$ and the short-term interest encoder $F_{short}(\cdot)$ follow directly from the architectures introduced in Chapter 3. Specifically, the computation of the long-term representation e_{long} corresponds to the process described in Section 3.2.4 and is formally defined in Equation 3.18. The short-term representation e_{short} is derived according to the design presented in Section 3.2.5 and mathematically specified in Equation 3.27.

The resulting embeddings e_{long} and e_{short} capture user interests at different temporal scales. Specifically, e_{long} summarizes the user's long-term interests, reflecting stable and relatively persistent preferences accumulated across the entire interaction history, such as favored categories or consistently preferred item types. In contrast, e_{short} encodes short-term interests, which represent the user's most recent and rapidly changing intents influenced by contextual factors like recency, session-specific goals, or transient trends. These two embeddings provide complementary views of user behavior and serve as the inputs to the diffusion-based fusion module.

4.2.3 Diffusion-based Fusion Module

The diffusion-based fusion module forms the core of DiffLSRec, enabling a generative mechanism for integrating long-term and short-term user interests. Rather than fusing behavioral signals through a single-step operation such as weighted averaging, gating, or attention pooling, this module views representation fusion as a progressive multi-step reconstruction process. The key idea is to treat the long-term embedding as a stable preference prior and progressively adds noise to it through a forward diffusion process, where Gaussian noise is gradually injected across multiple timesteps. This controlled degradation allows the model to detach from rigid long-term patterns and increase representational flexibility.

During the reverse denoising stage, the short-term representation serves as a conditional guidance signal that incrementally steers the reconstruction of the noised latent state back toward a refined user representation. This iterative refinement framework enables deeper interaction between long- and short-term interests compared with conventional one-shot fusion strategies. Instead of merging signals uniformly, the diffusion process adaptively modulates how much influence short-term intent should exert at different stages of reconstruction, allowing the model to maintain stable historical preferences at early steps while incorporating fine-grained recent behaviors as uncertainty decreases. As a result, the diffusion-based fusion captures nuanced preference evolution and improves robustness under noisy or sparse sequential patterns.

This module is organized into two main operational stages. The first stage, **Forward Diffusion** 4.2.3, gradually corrupts the long-term representation by injecting Gaussian noise according to a predefined variance schedule. By doing so, the model systematically weakens dominant long-term signals and produces a latent trajectory that can be progressively reconstructed. The second stage, **Reverse Denoising** 4.2.3, performs step-by-step reconstruction of the latent representation by conditioning on the short-term intent. This stage is further enhanced with Token-level Contextual Enhancement, which extracts fine-grained sequential cues via cross-attention over recent interaction tokens, and Monotonic SNR-Adaptive Guidance, which dynamically regulates the contributions of long- and short-term signals based on the signal-to-noise ratio at each denoising step. Together, these mechanisms enable a smooth and interpretable transition from stable, history-driven priors to context-aware short-term adjustments.

Through this design, the diffusion-based fusion module provides a flexible and multi-stage mechanism for interest integration, enabling DiffLSRec to generate user representations that accurately reflect both persistent preferences and rapidly evolving behavioral signals.

Forward Diffusion

After obtaining the long-term embedding $e_{long} \in \mathbb{R}^d$ and the short-term embedding e_{short}, DiffLSRec formulates the fusion as a generative denoising process. In the forward stage, a noising path is constructed by gradually injecting Gaussian noise into the long-term representation. This step provides the basis for reverse denoising, where short-term signals can be injected later to improve the corrupted representation. Starting from the stable prior e_{long} that encodes a user's long-term interests, the process produces a sequence of latent variables $\{e_t\}_{t=1}^{T}$ with progressively higher noise levels. This progressive noising creates a multi-step pathway that enables short-term interests to influence the representation step by step, rather than through a single fusion operation. Adding stochasticity by noising also serves as regularization, preventing the model from depending too heavily on a fixed deterministic vector. The diffusion path shows a coarse-to-fine progression, where earlier steps preserve more global preference information, while later steps are increasingly dominated by noise. This design improves robustness to distribution shifts and allows the model to adaptively balance the strength of long- and short-term signals across different timesteps.

The forward diffusion process is modeled as a first-order Markov chain, where each latent variable e_t is obtained by gradually injecting Gaussian noise into the previous state e_{t-1}. This construction ensures that the generation of e_t depends only on its immediate predecessor, instead of the entire history, which simplifies both the modeling and inference process. Through multiple timesteps, this Markovian process generates a sequence of latent variables latent variables $\{e_t\}_{t=1}^{T}$, where each state is a noisier transformation of the original long-term embedding *elong*.

$$q(e_t \mid e_{t-1}) = \mathcal{N}(\sqrt{\alpha_t} e_{t-1}, (1-\alpha_t) I), \quad e_0 = e_{long} \tag{4.4}$$

where $\mathcal{N}(\mu, \Sigma)$ denotes a multivariate Gaussian distribution with mean vector μ and covariance matrix Σ. t denotes the timestep. The parameter $\alpha_t \in (0, 1)$ is a step-dependent noise-schedule coefficient that controls how much signal is retained at timestep t. I is the $d \times d$ identity matrix to ensure isotropic noise. The process starts with $e_0 = e_{\text{long}}$, the long-term embedding that serves as the prior. The input of this equation is the previous latent e_{t-1}, and the output is the conditional distribution over the noised embedding e_t.

By repeatedly applying the above step-wise equation, the distribution for computing e_t from the original long-term embedding e_{long} can be directly expressed as a closed-form marginal distribution without explicitly simulating all intermediate steps:

$$q(e_t \mid e_{long}) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\, e_{long},\, (1-\bar{\alpha}_t) I), \quad \bar{\alpha}_t = \prod_{s=1}^{t} (1-\beta_s) \tag{4.5}$$

where β_s is the noise coefficient at step s and $\bar{\alpha}_t$ is the cumulative product of noise scheduling coefficients, and $\mathcal{N}(\mu, \Sigma)$ denotes a multivariate Gaussian distribution. The input is e_{long}, and the output is the distribution describing e_t at timestep t.

To convert the probabilistic distribution into an actual vector that can be fed into the model, the noisy latent e_t is sampled using the reparameterization trick. Equation 4.5 defines the Gaussian distribution of the noisy latent, while Equation 4.6 provides its reparameterized sampling form, which is mathematically equivalent but enables differentiable training in neural networks. In this equation, the deterministic part depends on the original long-term embedding e_{long}, while the stochastic part depends only on Gaussian noise ϵ. This design enables gradients to propagate back to e_{long} and parameters, while the randomness does not block gradient flow:

$$e_t = \sqrt{\bar{\alpha}_t}\, e_{long} + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{4.6}$$

where ϵ denotes Gaussian noise, $\bar{\alpha}_t$ is the cumulative product of noise scheduling coefficients, and I is the matrix to ensure isotropic noise.

Through this forward diffusion process, the long-term representation is transformed into a progressively noisier latent sequence $\{e_t\}_{t=1}^{T}$, with e_T denoting the final maximally corrupted state. This terminal latent serves as the starting point for the reverse denoising phase, where short-term signals are injected to guide the reconstruction of the clean representation.

Reverse Denoising

In the reverse stage, the goal is to gradually recover the clean representation from the noisy latent produced by the forward process, while combining short-term interests as conditional guidance. The denoising begins from the maximally corrupted latent e_T, which is the final element of the noisy sequence $\{e_t\}_{t=1}^{T}$ obtained during forward diffusion. Starting from e_T, the model repeatedly removes noise. For $t = T, \ldots, 1$ it outputs e_{t-1}, and the sequence $\{e_{t-1}\}_{t=1}^{T}$ progressively reconstructs the long-term embedding. This gradual recovery enables short-term interests to be injected during the denoising process, allowing them to shape the final representation rather than being introduced only once at the end. In the early steps, removing large-scale noise helps quickly restore the global structure of the representation, while the later steps focus on fine-grained adjustments that refine local details. This staged denoising strategy balances the stability provided by long-term interests with the adaptability by short-term signals.

The reverse denoising process is also modeled as a first-order Markov chain, meaning that the generation of each latent variable e_{t-1} depends only on the current latent e_t and the short-term embedding e_{short}, rather than the entire history. This design simplifies the modeling process, while allowing the model to capture how the current state evolves under the influence of short-term signals. Each denoising step is defined as sampling from a conditional Gaussian distribution:

$$p_\theta(e_{t-1} \mid e_t, e_{short}, S_{short}) = \mathcal{N}(\mu_\theta(e_t, t, e_{short}, S_{short}), \sigma_t^2 I) \tag{4.7}$$

where $p_\theta(\cdot)$ denotes the reverse transition distribution parameterized by a neural network with parameters θ. The variable e_t is the noisy latent representation at timestep t, and e_{t-1} is its denoised counterpart at the previous step. The conditional input e_{short} represents the short-term embedding that guides the denoising process. The variance term $\sigma_t^2 I$ is a diagonal covariance matrix, where σ_t^2 is a predefined variance schedule and I is the identity matrix ensuring isotropic noise. $\mathcal{N}(\cdot, \cdot)$ denotes a multivariate Gaussian distribution.

Token-level Contextual Enhancement While the aggregated short-term embedding e_{short} summarizes the user's recent intent, it may lose the fine-grained contextual information among the most recent interactions. To better capture sequential dependencies, token-level contextual enhancement is introduced, which incorporates the embeddings of the most recent k items $S_{\text{short}} = [e_{T-k+1}, e_{T-k+2}, \ldots, e_{T-1}]$ as conditional tokens during the reverse denoising process.

At each denoising step t, the noisy latent representation e_t interacts with these contextual tokens through a cross-attention mechanism:

$$C_t(e_t, S_{short}) = \text{softmax}\left(\frac{(W_Q e_t)(W_K S_{short})^\top}{\sqrt{d}}\right)(W_V S_{short}) \tag{4.8}$$

where W_Q, W_K, and W_V are the projection matrices for queries, keys, and values. The resulting representation C_t captures token-level context and highlights the most relevant recent interactions.

To adaptively balance the token-level context C_t and the aggregated short-term vector e_{short}, a gating mechanism is introduced:

$$c_t = \sigma(W_g[e_t; e_{short}]) \cdot C_t + (1 - \sigma(W_g[e_t; e_{short}])) \cdot e_{short} \tag{4.9}$$

where W_g is a learnable gating parameter and $\sigma(\cdot)$ is the sigmoid activation. The fused conditional embedding c_t serves as the enhanced guidance for the denoising network, replacing the original short-term condition e_{short} in the noise prediction function ε_θ.

Monotonic SNR-Adaptive Guidance Although conditional reverse denoising allows the aggregated short-term representation e_{short} and token-level contextual tokens S_{short} to refine the long-term prior, relying on a fixed guidance scale may lead to either weak or overly rigid short-term influence. In real recommendation scenarios, the importance of short- and long-term interests evolves dynamically. For instance, users may consistently follow habitual behaviors (e.g., repurchasing familiar brands) or occasionally shift attention to new goals such as exploring trending items.

To capture variability while avoiding unstable fusion, we extend diffusion-based classifier-free guidance with a **monotonic Signal-to-Noise Ratio–aware Adaptive Conditional Guidance (ACG-SNR)** mechanism that adjusts guidance intensity according to the diffusion stage and contextual uncertainty. The monotonic design explicitly constrains how short-term guidance is introduced, preventing noisy short-term behaviors from having excessive influence at early stages—where representations are highly uncertain and noise-dominated—and from overwhelming the long-term prior before sufficient refinement, especially under sparse or noisy interaction data.

Specifically, the incorporation of the signal-to-noise ratio (SNR) enables the model to assess the uncertainty level of the current latent state. When uncertainty is high, long-term preferences are emphasized to provide a stable global prior that constrains the denoising trajectory. As noise gradually decreases in later diffusion steps, short-term behaviors—encoding high-frequency and context-specific interests—are progressively incorporated to refine the representation toward the user's current intent. This monotonic transition ensures a stable and adaptive fusion process, balancing robustness against noise with sensitivity to recent behavioral signals.

To make this transition stable, a **monotonicity constraint** is imposed on the guidance scale s_t, ensuring that it increases smoothly as the noise level drops (i.e., as SNR rises). Without this constraint, s_t might fluctuate or even decrease temporarily, leading to unstable denoising trajectories and semantic inconsistencies. These designs help ACG-SNR function as a "smart controller" that understands when to trust global long-term preferences and when to emphasize local short-term behaviors, resulting in steadier optimization and more realistic user-interest reconstruction.

At each denoising step, two versions of the predicted noise are obtained: the conditional prediction $\epsilon_\theta(e_t, t \mid c_t)$, which incorporates both e_{short} and S_{short} as contextual guidance, and the unconditional prediction $\epsilon_\theta(e_t, t)$, which depends solely on the long-term prior. These two represent opposite extremes of pure short-term or pure long-term influence. To balance them adaptively, ACG-SNR interpolates between the two predictions using a time-

and state-dependent scale s_t:

$$\hat{\epsilon}_\theta^{\text{ACG-SNR}} = (1 + s_t)\, \epsilon_\theta(e_t, t \mid c_t) - s_t\, \epsilon_\theta(e_t, t \mid \varnothing), \tag{4.10}$$

where e_t is the noisy latent at diffusion step t, and $\epsilon_\theta(\cdot)$ denotes the noise prediction network parameterized by θ. The adaptive guidance scale $s_t \in [0, s_{\max}]$ is computed as:

$$s_t = s_{\max} \cdot \sigma\Big(- a\big(\tfrac{t}{T} - b\big) + \mathbf{u}^\top [\, \|e_t\|,\, \|e_{short}\|,\, \log(1 + \text{SNR}_t)\,] + b_0 \Big) \tag{4.11}$$

where $\sigma(\cdot)$ is the sigmoid activation that bounds s_t within $[0, s_{\max}]$. $s_{\max}$ is the maximum adaptive guidance scale, which determines the largest possible influence of short-term interests during denoising. $a > 0$ denotes the slope of the monotonic logistic time prior determining how quickly the transition occurs. b is the offset of the logistic prior, specifying at which normalized diffusion step the guidance shifts from strong long-term to strong short-term influence. t is current diffusion step index and T is the total number of diffusion steps. $\mathbf{u}^\top[\cdot]$ is the learnable linear projection that adaptively modulates s_t according to the contextual information, where $\|e_t\|$ denotes the magnitude of the current noisy latent, $\|e_{short}\|$ measures the strength of the short-term preference. $-a\big(\frac{t}{T} - b\big)$ defines a monotonic logistic time prior that ensures s_t evolves smoothly and monotonically with the diffusion step t. $\log(1 + \text{SNR}_t)$ explicitly encodes the signal-to-noise ratio $\text{SNR}_t = \frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}$ at step t, while $\bar{\alpha}_t$ denotes cumulative product same as Equation 4.12.

This design allows the model to dynamically balance long-term stability and short-term adaptability. At early reverse-diffusion steps, where the SNR is low and uncertainty dominates, s_t remains small, ensuring that long-term preferences provide a stable global prior and preventing noisy short-term signals from having excessive influence. As the denoising progresses and the SNR increases, s_t rises monotonically, strengthening short-term contextual guidance and refining the generated preferences once sufficient uncertainty has been removed. ACG-SNR enables an interpretable and controllable interest fusion process across the denoising trajectory, explicitly regulating when short-term behaviors should intervene. The model learns when to rely on habits and when to react to change—it maintains the user's general intent when behavior is stable, and flexibly captures instant interests when their preference shifts. This structured adaptive balancing not only improves recommendation accuracy but also enhances personalization by aligning the generated items with users' evolving preferences.

To obtain a stable reverse denoising trajectory, we recover the clean latent representation at each step by directly removing the predicted noise from the current noisy state e_t. Instead of modeling the reverse transition distribution explicitly, we estimate the clean latent $\hat{e}_0$ in a single computation by subtracting the ACG-SNR–guided noise prediction ϵ_θ

from e_t. This design prevents error accumulation across iterative updates, produces a numerically consistent refinement of user representations, and maintains the balance between long-term semantics and short-term contextual information. The resulting clean estimate is formulated as follows:

$$\hat{e}_0(e_t, t, e_{\text{short}}, S_{\text{short}}) = \frac{e_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\epsilon}_\theta^{\text{ACG-SNR}}}{\sqrt{\bar{\alpha}_t}} \quad \bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s) \tag{4.12}$$

where β_t is the noise coefficient at step t, and $\bar{\alpha}_t$ denotes its cumulative product.

This formulation gives us a stable estimate of the user's latent intent at each step, allowing the reverse process to progress smoothly without sudden drifts in the semantic space. By explicitly constructing the clean estimate $\hat{e}_0$, the model starts each reverse step from a clearer and more reliable state, which makes the entire denoising procedure more robust and easier to control.

During inference, the model gradually denoises the latent representation by updating e_t from $t = T$ down to $t = 1$. Each reverse step constructs e_{t-1} by mixing the clean estimate $\hat{e}_0$, the ACG-SNR–guided noise prediction, and an optional Gaussian term that injects mild randomness. This produces a smooth denoising trajectory that eventually converges to e_0, which is treated as the final clean representation.

$$e_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{e}_0(e_t, t, e_{\text{short}}, S_{\text{short}}) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\,\hat{\epsilon}_\theta^{\text{ACG-SNR}} + \sigma_t z \tag{4.13}$$

where the term $\sigma_t z$ injects a small amount of Gaussian noise, with z to introduce controlled stochasticity during the reverse update. This optional noise term is controlled by the variance schedule σ_t.

The model is trained by minimizing the mean squared error between the true noise ϵ and the predicted noise ϵ_θ. This loss allows the network to accurately predict the noise at each step, enabling the long-term embedding to be progressively refined under the guidance of short-term signals:

$$\mathcal{L}_{denoise} = \mathbb{E}_{t, e_{long}, \epsilon, u}\left[\left\|\epsilon - \epsilon_\theta\Big(\sqrt{\bar{\alpha}_t}\,e_{long} + \sqrt{1-\bar{\alpha}_t}\,\epsilon,\ t \,\Big|\, c_t\Big)\right\|^2\right] \tag{4.14}$$

where $\mathbb{E}$ denotes the expectation operator, and $\|\cdot\|$ represents the Euclidean norm. c_t is the token-level enhanced representation used as the conditioning variable for classifier-free guidance, applied with probability $1-p$ for conditional case and $\varnothing$ with probability p for unconditional case.

To improve the stability and generalization of the denoising process, L2 regularization is used in the diffusion-based fusion network. L2 weight regularization is applied to the projection matrices W_Q, W_K, W_V, the gating parameters W_g, and the feed-forward layers in the denoising network. The overall training objective is formulated as follows:

$$\mathcal{L}_{diff} = \mathcal{L}_{denoise} + \lambda_{reg}\big(\|W_Q\|_2^2 + \|W_K\|_2^2 + \|W_V\|_2^2 + \|W_g\|_2^2\big) \tag{4.15}$$

where λ_{reg} denotes the regularization coefficient controlling the strength of weight decay.

As conditioning is applied repeatedly across multiple steps, the fusion becomes deeper and more flexible than single-step operators. The model is able to emphasize short-term signals when they are more important and rely on long-term signals in more stable situations. The reverse process provides e_0 as the denoised representation, which is serves as the input to the prediction layer.

4.2.4 Final Prediction

Overall, the diffusion-based fusion module defines a function F_θ that maps long- and short-term interests into a fused representation. After T denoising steps by integrating both stable long-term interests and rapidly evolving short-term interests, the final fused user representation is obtained as

$$e_{user} = e_0 = F_\theta(e_{\text{long}}, e_{\text{short}}, S_{\text{short}}) \tag{4.16}$$

This denoised embedding e_{user} reflects a progressive refinement process where noise is gradually removed while short-term signals are injected through classifier-free guidance. Compared to conventional fusion methods that rely on static weights or gates, this approach provides a more flexible and adaptive integration: long-term signals ensure consistency and robustness, while short-term signals allow rapid adaptation to immediate goals. The controllability introduced by the guidance scale s makes it possible to dynamically emphasize either long-term or short-term signals depending on the scenario, offering robustness for routine behaviors, adaptability for goal-driven sessions, and better support for cold-start users. The fused representation e_{user} captures a comprehensive view of user interests and is then fed into the prediction layer for personalized recommendation.

In the prediction stage, the fused user representation $e_{user} = e_0$ obtained from the diffusion-based fusion process is mapped into the item embedding space to compute recommendation scores. Each candidate item is represented by its embedding e_i, and the relevance between the user and the item is measured through their interaction. This step

connect the generative fusion of long- and short-term interests with the final ranking process, enabling the model to provide personalized recommendations over the entire item set.

To generate recommendation scores, each candidate item i is mapped to a trainable embedding vector $e_i \in \mathbb{R}^d$. The interaction score with user u is obtained by projecting e_{user} onto e_i through a dot-product operation.:

$$\hat{y}_{u,i} = e_{user}^{\top} e_i \tag{4.17}$$

which denotes the similarity between the user's fused representation and the item embedding in the latent space. A higher score $\hat{y}_{u,i}$ reflects a stronger relevance between the user and the item, meaning that user u is more likely to interact with item i.

To transform the raw scores into a probability distribution, a softmax function is applied over the entire item set:

$$P(i \mid u) = \frac{\exp(\hat{y}_{u,i})}{\sum_{j \in \mathcal{I}} \exp(\hat{y}_{u,j})} \tag{4.18}$$

where $\mathcal{I}$ denotes the set of all items. This equation allows the model to rank items by relevance and ensures that the probabilities across all items sum to one, which is suitable for next-item prediction tasks.

The training objective is to maximize the likelihood of the observed interactions while minimizing the probability assigned to unobserved items. The cross-entropy loss is used to penalize the model when the predicted probability of the ground-truth item is low:

$$\mathcal{L} = -\sum_{(u,i) \in \mathcal{D}} \log P(i \mid u) \tag{4.19}$$

where $\mathcal{D}$ denotes the set of user–item pairs in the training dataset. By minimizing this loss, the model encourages e_{user} to stay close to the embeddings of items the user has interacted with, while further away from items that are not relevant.

Through this prediction layer, the denoised user representation obtained from the diffusion-based fusion module is connected to the recommendation task. The use of dot-product similarity coupled with the softmax normalization ensures computational efficiency and compatibility with widely adopted sequential recommendation frameworks. This consistency enables a fair comparison with representative baselines, while highlighting the benefits of generative fusion strategy of DiffLSRec in generating more context-aware and adaptive user representations.

4.3 Experimental Evaluation

This section presents the experimental evaluation of DiffLSRec and examines its effectiveness under realistic sequential recommendation settings. The goal of this chapter is to understand how the proposed diffusion-based generative fusion mechanism influences recommendation performance, how well DiffLSRec competes against existing architectures, and how different components contribute to the overall behavior of the model. To enable a fair comparison with the analysis in Chapter 3, the same Amazon Reviews 2023 dataset and the same three representative domains—`Cell_Phones_and_Accessories`, `Movies_and_TV`, and `Clothing_Shoes_and_Jewelry`—are used throughout the experiments. These domains span contrasting interaction dynamics, from rapid item turnover to stable genre-driven patterns, making them suitable for evaluating both generative refinement of long-term preferences and conditional guidance from short-term intents.

DiffLSRec is benchmarked against several representative sequential recommendation baselines—Fossil, MARank, GRU4Rec, HGN, and SASRec—to evaluate how well the diffusion-based formulation improves next-item prediction relative to Markov-based, multi-order attention, recurrent, gating-based, and self-attention architectures. These baselines capture different modeling assumptions about user behavior sequences, allowing a comprehensive comparison of whether iterative denoising and multi-stage interest refinement can outperform conventional single-step fusion mechanisms.

Evaluation is conducted using four widely adopted top-k ranking metrics (HR@10, NDCG@10, MRR@10, and Precision@10), which together quantify retrieval accuracy, ranking quality, and position-sensitive relevance. Since diffusion-based models generate representations through progressive refinement, position-aware metrics such as NDCG@10 and MRR@10 are especially informative for assessing whether the denoising trajectory enhances the final recommendation ranking. By comparing DiffLSRec with diverse baselines under consistent datasets and domains, the experiments aim to validate the effectiveness of generative multi-step fusion, token-level contextual enhancement, and SNR-guided interest balancing.

To examine how the representation capacity of DiffLSRec influences its performance, we vary the embedding dimension $\mathbf{K}$ in the range $\{8, 16, 32, 64, 128\}$. The embedding dimension controls the size of the latent vectors used to encode users and items. While a larger $\mathbf{K}$ provides a more expressive representation space, it also increases the number of parameters and may introduce overfitting, especially on relatively small datasets. By observing the performance trend across different values of $\mathbf{K}$, we can assess the stability of DiffLSRec and its sensitivity to this key architectural hyperparameter.

In addition to analyzing the impact of embedding dimensionality, we also conduct an ablation study to understand how each part of DiffLSRec contributes to its overall performance. We first compare our diffusion-based generative fusion with a traditional gate-based fusion baseline to test whether the multi-step denoising process yields measurable benefits over static fusion. Then we further examine three simplified variants of DiffLSRec: removing the Token-level Contextual Enhancement (w/o Token), replacing the Monotonic SNR-Adaptive Guidance with conventional classifier-free guidance (w/o SNR), and removing both components simultaneously (w/o Both). These comparisons allow us to quantify the effect of each enhancement module and verify that the progressive generative fusion, together with contextual refinement and SNR-aware balancing, collectively contributes to the performance improvements observed in DiffLSRec.

4.3.1 Dataset

The experiments for DiffLSRec are conducted using the same Amazon Reviews 2023 dataset introduced earlier in Section 3.3.1. This dataset provides large-scale user–item interaction histories collected from May 1996 to September 2023, with over 571 million records and a broad coverage of product domains. Its fine-grained timestamps, consolidated product families, and standardized metadata make it especially suitable for evaluating generative sequential models that rely on temporal dynamics and stable long-term preference priors.

For DiffLSRec, the dataset is used without modifying the domain definitions in Chapter 3. Three representative categories—`Cell_Phones_and_Accessories`, `Movies_and_TV`, and `Clothing_Shoes_and_Jewelry`—are again selected to examine model behavior across domains with different long-short preference structures. These domains present contrasting patterns: rapid item turnover and strong recency effects in Cell Phones, stable genre-driven preferences in Movies, and extremely high sparsity and long-tail distributions in Clothing. Such domain diversity provides a comprehensive testing environment for analyzing how effectively DiffLSRec can fuse long-term and short-term behavioral signals.

Compared with the experimental setup in Chapter 3, a larger number of samples is used for training and evaluation in this chapter. As DiffLSRec operates through multi-step forward diffusion and reverse denoising, the model requires a broader range of behavioral patterns to stabilize the generative trajectory. Moreover, the incorporation of token-level contextual enhancement and SNR-guided refinement introduces additional learnable interactions that benefit from more training instances. For these reasons, we increase the sample size beyond the 10,000 interactions used for TiIfSRec, enabling more reliable optimization of the diffusion-based fusion mechanism.

To better understand the characteristics of the dataset, we perform basic data analysis and visualization, including the distribution of user interaction lengths, item frequency, and temporal intervals. These analyses help identify key properties such as sparsity, long-tail item distribution, and variability in user activity, which guide the design and evaluation of the proposed models. Due to the large scale of the original dataset, a subset is used for efficient experimentation. We select users with a minimum number of interactions to ensure meaningful sequential patterns, and limit the dataset size to maintain computational feasibility while preserving representative data characteristics. The final subset used in our experiments contains 100,000 interaction records.

The essential fields are extracted from the dataset, consistent with Section 3.3.1, including `user_id`, `timestamp`, `verified_purchase`, and `parent_asin`. The `parent_asin` field is used to group product variants into unified product families, ensuring consistent identity resolution when generating short-term context tokens and long-term priors. Titles and metadata associated with each `parent_asin` are also utilized for maintaining semantic consistency across diffusion steps.

4.3.2 Evaluation Metrics

DiffLSRec is evaluated by the following four common metrics to top-k recommendation performance. These evaluation metrics reflect different dimensions of ranking quality, enabling a comprehensive and standardized comparison among all baseline models.

- **HR@10**: Hit Ratio measures the proportion of users for whom the ground-truth item appears within the top 10 recommendations [7]. It reflects the overall success rate of the recommender system in predicting relevant items among the top-k ranked results.

$$\text{HR@10} = \frac{1}{N}\sum_{u=1}^{N}\mathbb{I}\big(\text{Hit@10}(u)\big) \tag{4.20}$$

  where $\mathbb{I}(\cdot)$ is an indicator function that equals 1 if the ground-truth item for user u is ranked within the top 10 predictions, and 0 otherwise.

- **NDCG@10**: NDCG is a position-aware metric which assigns larger weights on higher positions. It evaluates the ranking quality of relevant items in the recommendation list.

$$\text{NDCG@10} = \frac{\text{DCG@10}}{\text{IDCG@10}} = \frac{\sum_{i=1}^{10}\frac{rel(i)}{\log_2(i+1)}}{\sum_{i=1}^{10}\frac{rel^*(i)}{\log_2(i+1)}} \tag{4.21}$$

where $rel(i)$ is the relevance score for the i-th position, and $rel^*(i)$ is the relevance score of the i-th position in the ideal case.

- **MRR@10**: Mean Reciprocal Rank evaluates the ranking quality by considering the position of the first relevant item in the top 10 recommendations. It assigns higher scores when relevant items appear earlier in the ranked list.

$$\text{MRR@10} = \frac{1}{N} \sum_{u=1}^{N} \frac{1}{\text{rank}_u} \tag{4.22}$$

  where rank_u denotes the position of the first relevant item for user u within the top 10 recommendations, and N is the total number of users.

- **Precision@10**: It calculates how many of the top 10 recommendations are actual interested items for the user, averaged across all N users.

$$\text{Precision@10} = \frac{1}{N} \sum_{u=1}^{N} \frac{\textit{Number of relevant items in top10}}{10} \tag{4.23}$$

These four metrics provide a comprehensive assessment of DiffLSRec from multiple perspectives. HR@10 directly measures the ability of the model to retrieve the correct next item, offering an intuitive view of overall hit performance. NDCG@10 emphasizes the ranking quality at higher positions, making it particularly informative for diffusion-based models where iterative refinement affects ordering sensitivity. MRR@10 evaluates how early the correct item appears in the ranked list, reflecting whether the denoising process successfully pushes relevant items toward the top. Precision@10 complements these metrics by quantifying how many of the recommended items are actually relevant to the user. Overall, the combination of these measures enables a robust evaluation of DiffLSRec in terms of retrieval accuracy, ranking consistency, and position-aware relevance within top-k recommendation scenarios.

4.3.3 Baselines

To evaluate the effectiveness of the proposed DiffLSRec model, we compare it with several representative sequential recommendation baselines. All these methods are widely used in modeling user-item interaction sequences. Their key characteristics are summarized as follows:

- **Fossil** [6]: A hybrid model that fuses Markov Chains with similarity-based collaborative filtering to integrate long-term preferences and short-term transition patterns through a linear combination.
- **MARank** [43]: A multi-order attentive ranking model that captures diverse user interests across different temporal levels by applying attention over multiple interaction orders.
- **GRU4Rec** [9]: A session-based sequential recommendation model that employs Gated Recurrent Units (GRUs) to capture users' short-term interests from recent interaction sequences.
- **HGN** [23]: The Hierarchical Gating Network models both long- and short-term user preferences in parallel and adaptively balances them through a hierarchical gating mechanism.
- **SASRec** [14]: A Transformer-based model that applies self-attention to capture long-range dependencies within user behavior sequences, representing one of the first self-attention approaches in sequential recommendation.
- **DiffRec** [38]: A diffusion-based sequential recommendation framework that formulates next-item prediction as a denoising process, modeling item dependencies through generative noise reconstruction.

These baselines cover a broad spectrum of sequential recommendation paradigms, ranging from Markov-based modeling and recurrent architectures to attention-driven methods and recent diffusion-based generative approaches. They provide a diverse and representative reference set, enabling a rigorous evaluation of the performance gains contributed by DiffLSRec under different modeling assumptions and learning mechanisms. The set of baselines in Chapter 4 differs from that in Chapter 3 due to their distinct purposes. In Chapter 3, we select representative models such as GRU-based and attention-based methods to illustrate key architectural differences and motivate the design of the proposed model. In contrast, Chapter 4 includes a broader set of baselines covering additional modeling paradigms, including Markov-based methods (e.g., Fossil), gating-based models (e.g., HGN), and recent diffusion-based approaches (e.g., DiffRec), to ensure a more comprehensive and fair empirical evaluation.

4.3.4 Results and Analysis

In this section, we analyze the performance of DiffLSRec and compare it with several representative sequential recommendation baselines across three datasets. Beyond reporting overall improvements in top-k accuracy, we further examine under which data conditions and dataset characteristics these gains are most evident. In particular, we study how the diffusion-based long–short interest fusion contributes to performance across datasets with different sparsity levels and interaction patterns. By analyzing metric variations, cross-domain differences, and relative performance gaps between models, we aim to better understand the role of each modeling component and identify the scenarios where diffusion-based long–short fusion provides the benefits.

Table 4.1 reports the performance results on the three benchmark datasets. DiffLSRec surpasses all baseline methods on HR@10, NDCG@10, MRR@10, and Prec@10, where Prec@10 corresponds to the Precision@10 metric defined in Equation 4.23. Marked values is the best baseline for computing improvements. These results collectively indicate that the proposed diffusion-driven fusion of long- and short-term interests effectively enhances the prediction accuracy.

As shown in Table 4.1, DiffLSRec achieves the best overall performance across the Cellphone, Clothes, and Movies datasets. On all four evaluation metrics (HR@10, NDCG@10, MRR@10, and Precision@10), DiffLSRec consistently outperforms the strongest baseline, DiffRec, as well as traditional sequential recommendation models such as GRU4Rec, MARank, HGN, and SASRec. Although the absolute metric values on the Cellphone and Clothes datasets are relatively low due to their small size and sparse user–item interactions, DiffLSRec still demonstrates clear and stable improvements, indicating the robustness of the proposed approach.

Traditional sequential models show moderate performance across datasets, but they often rely on static hidden states or single-step fusion mechanisms. This limits their ability to capture rapid changes in user interests and makes them less effective when long-term and short-term behaviors differ. DiffRec, the diffusion-based baseline, performs notably stronger than most neural models, suggesting that generative denoising is an effective paradigm for modeling sequential dependencies. However, DiffRec merges user interests into a single representation and refines it uniformly during the denoising process, which restricts its ability to adapt to more nuanced preference transitions.

In contrast, DiffLSRec introduces a multi-step long-short interest fusion mechanism, in which short-term interaction signals progressively guide the refinement of long-term user preference representations throughout the denoising trajectory. This design allows

Table 4.1: Result of experiment on DiffLSRec

Dataset	**Method**	**HR@10**	**NDCG@10**	**MRR@10**	**Prec@10**
Cellphone	Fossil	0.14200000	0.06752663	0.03854444	0.01420000
	MARank	0.16266666	0.07918197	0.04461111	0.01626667
	GRU4Rec	0.22333333	0.10486140	0.05764259	0.02233333
	HGN	0.19933334	0.09574587	0.05421772	0.01993333
	SASRec	0.22733334	0.12611172	0.04825476	0.02273333
	DiffRec	0.27866667	0.15196856	0.06194497	0.02786667
	DiffLSRec	**0.29999999**	**0.16195426**	**0.06867434**	**0.02999999**
	Improvement	**7.65%**	**6.57%**	**10.86%**	**7.65%**
Clothes	Fossil	0.15733333	0.09125891	0.03220820	0.01573333
	MARank	0.16400000	0.09664540	0.03752249	0.01640000
	GRU4Rec	0.17333333	0.12531130	0.03099180	0.01733333
	HGN	0.16599999	0.11025910	0.03642751	0.01660000
	SASRec	0.18000001	0.12487048	0.03665503	0.01799999
	DiffRec	0.19066666	0.12571591	0.03780634	0.01906666
	DiffLSRec	**0.20733332**	**0.13277971**	**0.04104312**	**0.02073333**
	Improvement	**8.73%**	**5.62%**	**8.57%**	**8.73%**
Movies	Fossil	0.57002109	0.45017547	0.08733845	0.05700212
	MARank	0.62702322	0.52075344	0.08343806	0.06270233
	GRU4Rec	0.69950742	0.65578878	0.08368687	0.06995074
	HGN	0.72836030	0.67007812	0.08389436	0.07283604
	SASRec	0.74876845	0.67094975	0.09523251	0.07487685
	DiffRec	0.75158340	0.67806209	0.08949795	0.07515835
	DiffLSRec	**0.77058411**	**0.68713196**	**0.09086464**	**0.07705842**
	Improvement	**2.53%**	**1.34%**	**1.53%**	**2.53%**

the model to adjust the contribution of different behavioral patterns at each diffusion step, leading to a more flexible and expressive user representation. As a result, DiffLSRec achieves noticeable improvements over DiffRec across all datasets (e.g., 7.65% HR@10 in Cellphone, 8.73% HR@10 in Clothes, and 2.53% HR@10 in Movies).

Even in the Movies dataset, where baseline models already achieve high performance due to denser interactions, DiffLSRec still has high performance. This indicates that the proposed fusion strategy remains beneficial even when user preferences are less sparse and sequential patterns are more stable.

Overall, the results demonstrate that the multi-step refinement of long-term and short-term user interests enables DiffLSRec to more effectively capture dynamic behavioral patterns, leading to improved recommendation quality across all tested datasets. The consistent improvements highlight the advantages of introducing diffusion-based fusion into sequential recommendation systems.

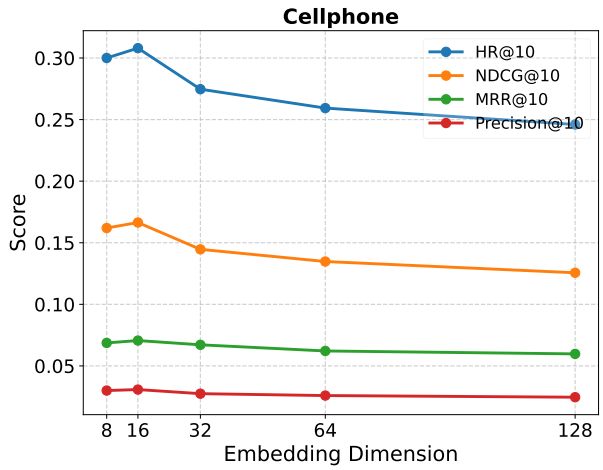
Figure 4.2 reports the performance of DiffLSRec under different embedding dimensions K on the three datasets. Overall, the results show a consistent pattern: smaller embedding sizes (8 or 16) lead to the best performance, while large dimensions (64 and 128) tend to reduce the performance.

On the Cellphone and Clothes datasets, which are relatively small and sparse, DiffLSRec achieves its highest HR@10, NDCG@10, MRR@10, and Precision@10 at $K = 8$ or $K = 16$. As K increases beyond 16, all four metrics steadily decrease. This indicates that, under limited data, excessively large embedding spaces introduce unnecessary parameters and cause overfitting and poor generalization.

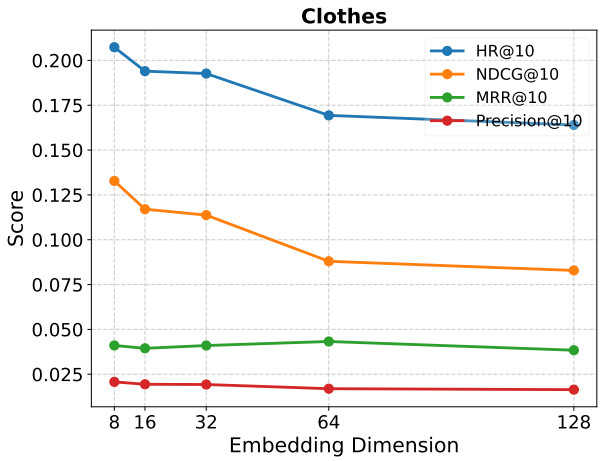
A similar tendency can be observed on the Movies dataset. DiffLSRec get the best NDCG@10 at $K = 8$, while HR@10 and MRR@10 peak at $K = 16$. For larger dimensions ($K \geq 32$), the performance on all metrics gradually declines. Although the absolute scores on Movies are higher due to richer interactions, the relative trend with respect to K remains the same as in the other two datasets.

Across all three datasets, the curves change smoothly and there are no abrupt drops, indicating that DiffLSRec is not overly sensitive to the exact choice of K as long as it stays in a reasonable range. This ablation study shows that a compact embedding space is sufficient for capturing user preferences in our setting, and that overly large dimensions mainly increase model capacity without bringing additional benefits.

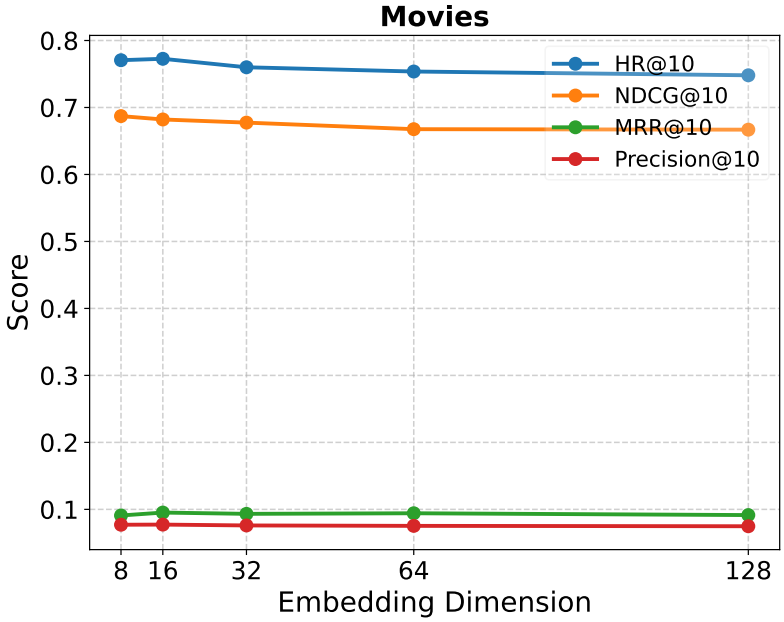
Furthermore, to verify the contribution of each component in DiffLSRec, we conducted an ablation study by selectively removing or modifying key modules. Four model variants were compared: a gate-based fusion baseline, a version without Token-level Contextual Enhancement (w/o Token), a version that replaces the SNR-adaptive guidance with standard



(a) Effect of embedding dimension on DiffLSRec in CellPhone

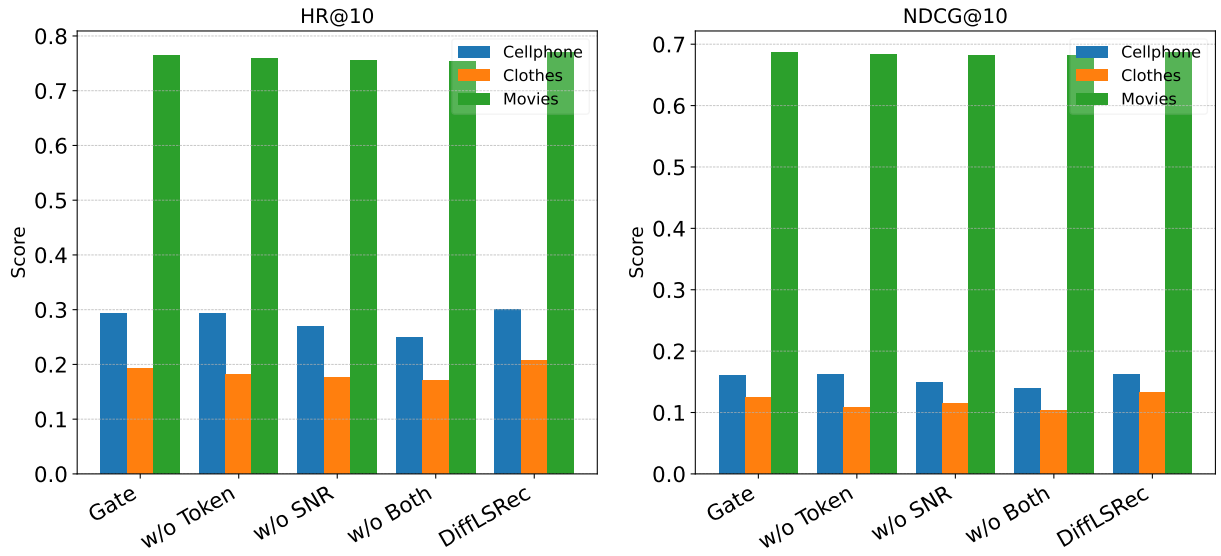


(b) Effect of embedding dimension on DiffLSRec in Clothes



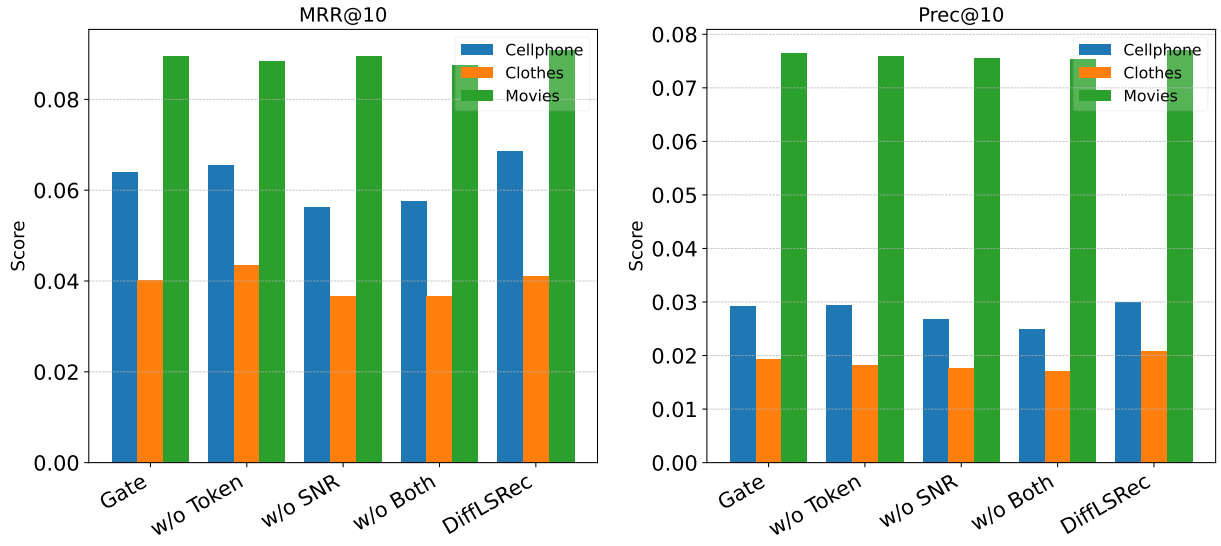
(c) Effect of embedding dimension on DiffLSRec Movies

Figure 4.2: Effect of the embedding dimension K on DiffLSRec across three domains



(a) HR@10 comparison in DiffLSRec ablation variants

(b) NDCG@10 comparison in DiffLSRec ablation variants



(c) MRR@10 comparison in DiffLSRec ablation variants

(d) Precision@10 comparison in DiffLSRec ablation variants

Figure 4.3: Ablation study of DiffLSRec and its variants

Table 4.2: Relative improvement of DiffLSRec over the Gate variant on three domains.

Metric	Cellphone	Clothes	Movies
HR@10	2.51%	7.24%	0.83%
NDCG@10	1.28%	6.55%	0.03%
MRR@10	7.52%	2.34%	1.57%
Precision@10	2.51%	7.24%	0.83%

classifier-free guidance (w/o SNR), and a version removing both enhancements simultaneously (w/o Both).

The results in Figure 4.3 reveal clear and consistent trends across all three datasets. Overall, the complete diffusion-based DiffLSRec achieves the best performance across all evaluation metrics. Removing either enhancement module leads to a noticeable decline, while disabling both results in the most significant degradation. The gate-based baseline performs moderately but still falls short of the diffusion-based fusion. These results demonstrate that each component contributes meaningfully to the model's effectiveness, and together they form a strong and complementary design that drives the superior performance of DiffLSRec.

Across the diffusion-based variants, removing the Token-level Contextual Enhancement generally leads to a clear decline in performance, particularly on the Cellphone and Clothes datasets. This indicates that the short-range contextual signals extracted from the most recent user interactions are especially valuable when user histories are relatively sparse or noisy. In contrast, the impact is milder on the Movies dataset, where longer interaction sequences already provide richer behavioral cues.

The SNR-adaptive guidance also plays an important role. When it is replaced by standard classifier-free guidance (w/o SNR), the performance decreases consistently across all datasets. The drop in HR@10 and Precision@10 on the Cellphone dataset suggest that the adaptive adjustment of the refinement strength helps stabilize the generative update steps and produces more reliable representations.

Removing both components at the same time leads to the greatest performance degradation. This version is consistently the weakest among the diffusion-based variants, supporting the view that the two enhancements are complementary: Token-level modeling improves contextual understanding, while SNR-guidance ensures a more stable and effective refinement trajectory.

The gate-based fusion baseline performs at a reasonable level, while it is less effective

Table 4.3: Comparison between TiIfSRec and DiffLSRec

Dataset	Model	HR@10	NDCG@10	MRR@10	Precision@10
Cellphone	TiIfSRec	0.2927	0.1599	0.0639	0.0293
	DiffLSRec	**0.3000**	**0.1620**	**0.0687**	**0.0300**
Clothes	TiIfSRec	0.1933	0.1246	0.0401	0.0193
	DiffLSRec	**0.2073**	**0.1328**	**0.0410**	**0.0207**
Movie	TiIfSRec	0.7643	0.6870	0.0895	0.0764
	DiffLSRec	**0.7706**	**0.6872**	**0.0909**	**0.0771**

than the diffusion-based variants. On the Cellphone and Movies datasets, its results are consistently lower than all diffusion versions, and on the Clothes dataset it is only similar to the w/o Token variant while still falling short of the full model. This is expected, because the gate mechanism combines long-term and short-term representations in a fixed, single-step way, without the multi-step refinement and flexibility that the diffusion process provides.

As shown in Table 4.2, DiffLSRec achieves consistent performance gains over the Gate variant across all three domains. The Gate variant corresponds to the fusion mechanism used in TiIfSRec. The results in Table 4.2 can be interpreted as the relative improvement of DiffLSRec over the TiIfSRec framework. The improvements are most pronounced on the Clothes dataset, indicating that the diffusion-based fusion mechanism is particularly effective in domains with high sparsity and rapidly shifting user interests. The Cellphone dataset exhibits moderate but stable improvements, suggesting that diffusion-based denoising still enhances ranking quality even when user behavior is relatively stable. In contrast, the Movies dataset shows smaller gains, likely due to longer interaction histories and more predictable viewing patterns. These results demonstrate that DiffLSRec is especially advantageous when short-term sequences are noisy, enabling more adaptive long–short interest integration than conventional gating.

Table 4.3 provides a direct comparison between the two core models of this thesis, TiIfSRec and DiffLSRec, under the same evaluation setting. Although Table 4.2 already reports the relative improvements of DiffLSRec over the deterministic gating-based fusion mechanism which corresponds to TiIfSRec, Table 4.3 presents the absolute performance values for a clearer side-by-side comparison.

As shown in Table 4.3, DiffLSRec consistently outperforms TiIfSRec on the Cellphone and Clothes datasets across all Top-10 metrics, although the magnitude of improvement

remains relatively small. The improvements are more noticeable on Clothes, where HR@10 and NDCG@10 increase from 0.1933 to 0.2073 and from 0.1246 to 0.1328, respectively. On Cellphone, the gains are smaller but consistent across all metrics. In contrast, on the Movie dataset, the two models achieve very similar performance. DiffLSRec slightly improves HR@10, MRR@10, and Precision@10, while TiIfSRec achieves a nearly identical NDCG@10 score.

These results suggest that DiffLSRec provides small but consistent improvements over TiIfSRec, and the advantage becomes more visible in datasets where user behavior is more irregular or sparse. When interaction patterns are more stable, as in the Movie dataset, the deterministic modeling bias in TiIfSRec already provides competitive performance and the additional flexibility of diffusion brings limited gains.

Overall, DiffLSRec achieves the strongest and most stable results. The consistent improvements across datasets highlight the combined benefits of contextual enhancement and guided refinement, and demonstrate the advantages of treating interest fusion as a generative process rather than relying on fixed or shallow combination rules.

4.4 Summary

DiffLSRec introduces a novel diffusion-based long–short interest fusion method that reframes sequential recommendation as a multi-step generative refinement process. Instead of compressing diverse user signals into a fixed gated architecture or performing a single-stage aggregation, DiffLSRec progressively enhances user representations through iterative denoising steps. At each step, short-term interaction provide adaptive and context-aware guidance to adjust the long-term preference state, allowing the model to gradually align short-lived behavioral variations with more stable historical patterns. This multi-step refinement mechanism captures subtle temporal dependencies that conventional recurrent or attention-based models often overlook. In addition, the incorporation of token-level contextual enhancement and an SNR-aware guidance schedule allows the model to control the strength of short-term signals dynamically, preventing noisy or highly sparse sequences from dominating too early in the generative process. As a result, DiffLSRec produces a more expressive, stable, and noise-tolerant characterization of user behavior.

Evaluations across three domains demonstrate the consistent advantages of this generative fusion strategy. DiffLSRec performs better than a wide range of traditional sequential recommenders as well as diffusion-style baselines that do not explicitly separate long- and short-term signals. The performance gains span multiple ranking metrics, including HR@10, NDCG@10, MRR@10, and Precision@10, highlighting robustness across

different evaluation perspectives. Moreover, the improvements are particularly evident in settings with rapid preference drift or high behavioral variability, where conventional models often struggle to maintain a balance between long-term stability and short-term adaptation. These findings confirm the effectiveness of treating sequential user modeling as a progressive generative denoising process rather than a one-pass deterministic encoding.

Although DiffLSRec achieves strong and stable performance, several limitations remain. The model can still be affected by data sparsity, especially in domains with many long-tail items or users who interact infrequently. In such cases, it may struggle to estimate reliable short-term embeddings, and early denoising errors can accumulate over the iterative refinement process. DiffLSRec also requires multiple inference steps, which increases computational cost compared to single-step models. This overhead is acceptable in most settings but may become a challenge in very large-scale or real-time applications without further optimization. In addition, because the model depends on patterns learned from interaction sequences, it may have difficulty handling highly irregular user behaviors or logs that lack enough contextual information to guide the diffusion steps. These limitations suggest directions for future work, including building more data-efficient training strategies and incorporating additional semantic or multimodal signals to reduce the impact of sparsity and improve early-stage refinement.

Overall, DiffLSRec provides a foundation for treating sequential recommendation as a generative refinement process, shifting the focus from single-step encoding to iterative representation improvement. By progressively integrating long- and short-term signals, the model offers a more flexible way to capture evolving user behavior and address the noise introduced by sparse or unstable sequences. The results in this chapter demonstrate that this formulation not only improves ranking performance across multiple datasets, but also generalizes well to different domains with varying interaction patterns. These findings highlight the potential of diffusion-based refinement as a promising direction for building more adaptive, stable, and data-efficient architectures in future research.

Chapter 5

Discussion

5.1 Introduction

This chapter provides a comprehensive discussion of the two proposed models, TiIfSRec and DiffLSRec, focusing on their methodological foundations, design rationales, strengths, limitations, and overall research value. Both models are developed to address fundamental challenges in sequential recommendation, particularly the modeling of long-term and short-term user interests under temporal dynamics, item frequency imbalance, and behavioral uncertainty. TiIfSRec and DiffLSRec are developed under a common research objective: to systematically address temporal dynamics, frequency imbalance, and multi-scale user interest modeling in sequential recommendation. TiIfSRec employs a deterministic dual-gated recurrent architecture to integrate temporal decay and frequency-aware adjustments in a unified framework, aiming for interpretability and computational efficiency. In contrast, DiffLSRec explores a diffusion-based probabilistic fusion mechanism to model complex interaction patterns and uncertainty in user behavior, enabling more flexible interest representation. Rather than viewing the two models as isolated contributions, this chapter considers them as complementary methodological explorations within a unified research framework. Through a structured discussion of their respective design choices and implications, this chapter aims to articulate the broader research significance of this work in advancing sequential recommendation modeling.

5.2 Strengths and Limitations of the Proposed Models

The proposed models, TiIfSRec and DiffLSRec, exhibit complementary strengths in modeling sequential user behavior by explicitly considering temporal intervals, item frequency imbalance, and uncertainty in interest evolution. Although both models are designed to jointly capture long-term and short-term user interests, they rely on different modeling principles. TiIfSRec adopts a structured and deterministic framework with clear inductive bias, whereas DiffLSRec introduces probabilistic generative modeling to enhance expressive flexibility. Together, they provide a more comprehensive view of interest modeling in sequential recommendation.

A major strength of TiIfSRec lies in its explicit integration of temporal and frequency signals into the representation learning process. The dual-gated mechanism embedded in the GRU-based long-term interest module enables the model to separately regulate temporal decay and frequency-aware adjustment. This separation allows the model to distinguish between recency effects and exposure bias, which are often entangled in conventional sequential models. By controlling these factors through learnable gates, TiIfSRec provides a structured way to model preference evolution over time while mitigating the dominance of highly frequent items. This design is particularly beneficial in scenarios where long-tail items play an important role and where user interests evolve gradually rather than abruptly. Another strength of TiIfSRec is its balance between modeling capacity and computational efficiency. Compared with fully attention-based architectures, the gated recurrent structure introduces a strong inductive bias that stabilizes learning in relatively sparse data conditions. The parameter scale and training complexity remain moderate, making the model more suitable for large-scale recommendation systems where inference latency and resource constraints must be considered. In addition, the explicit gating structure improves interpretability, as the learned gate values can provide insight into how temporal intervals and frequency signals influence user representation.

DiffLSRec enhances modeling flexibility by introducing diffusion-based probabilistic fusion between long-term and short-term interest representations. Rather than combining these components through fixed or deterministic weighting strategies, DiffLSRec treats their interaction as a stochastic generative process. This formulation allows the model to better capture uncertainty and variability in user behavior. In practical recommendation environments, user interests may fluctuate due to external factors, context shifts, or irregular activity patterns. The diffusion mechanism enables gradual refinement of the fused representation through iterative denoising, thereby capturing more complex interac-

tion patterns between historical preferences and recent behaviors. Furthermore, DiffLSRec demonstrates improved robustness in scenarios with noisy or sparse interaction sequences. By modeling interest fusion through a probabilistic framework, the model can mitigate the impact of occasional anomalous interactions or abrupt preference changes. This characteristic is particularly valuable in environments where user behavior is highly dynamic or where data quality is imperfect.

Overall, TiIfSRec and DiffLSRec represent two complementary modeling perspectives. TiIfSRec emphasizes structured modeling, interpretability, and deployment feasibility, while DiffLSRec focuses on flexibility, uncertainty modeling, and representational richness. They provide a broader and more nuanced approach to long- and short-term interest modeling in sequential recommendation systems. Despite their strengths, both models exhibit limitations that stem from their underlying design assumptions and modeling priorities.

For TiIfSRec, the reliance on deterministic gating mechanisms may restrict its ability to represent highly irregular or abrupt preference shifts. The model assumes that user interest evolves in a relatively structured manner that can be regulated through temporal decay and frequency adjustment. While this assumption holds in many real-world scenarios, it may be less effective when user behavior exhibits sudden contextual transitions or short-lived but strong interests. In such cases, deterministic fusion of long-term and short-term representations may not fully capture behavioral uncertainty. In addition, the effectiveness of the time-aware components in TiIfSRec depends on the availability and quality of temporal interval information. If timestamps are noisy, missing, or recorded at coarse granularity, the benefit of explicit interval modeling may be reduced. Although the gating mechanism provides robustness to some extent, inaccurate temporal signals can still influence representation learning.

For DiffLSRec, the main limitation lies in its increased computational cost and training complexity. The diffusion process requires iterative sampling steps and careful configuration of noise schedules and guidance parameters. As a result, training time and resource consumption are significantly higher compared to TiIfSRec. This characteristic may limit the model's practicality in real-time or resource-constrained production systems without additional optimization strategies. Moreover, diffusion-based modeling introduces additional hyperparameters that may require careful tuning to achieve stable performance. While this flexibility enhances expressive capacity, it also increases implementation complexity and sensitivity to configuration choices.

These differences reflect a fundamental trade-off between structured efficiency and probabilistic expressiveness. TiIfSRec prioritizes interpretability, stability, and computational practicality, while DiffLSRec prioritizes modeling flexibility and robustness under uncer-

tainty. Rather than representing weaknesses, these limitations highlight distinct modeling choices under different operational constraints. Recognizing these trade-offs clarifies the appropriate application scenarios for each model. It also suggests potential future directions, such as integrating lightweight stochastic components into structured gating frameworks or developing more computationally efficient diffusion strategies tailored to sequential recommendation tasks.

5.3 Discussion on Methodology and Design Decisions

This section explains the main design choices in TiIfSRec and DiffLSRec and the reasoning behind them. The focus is not on implementation details, but on why these modeling strategies are suitable for sequential recommendation tasks with temporal dynamics, frequency imbalance, and behavioral uncertainty.

For long-term interest modeling in TiIfSRec, a GRU-based recurrent structure is adopted. A recurrent model is a natural choice for sequential data because it updates the user representation step by step according to the chronological order of interactions. Compared with LSTM, GRU has a simpler gating mechanism and fewer parameters, which makes optimization easier and reduces computational cost while still being able to model long-range dependencies. In recommendation datasets where interaction sequences can be sparse and item vocabularies are large, this parameter efficiency is important to maintain stable training. Compared with purely attention-based architectures, a recurrent backbone also provides stronger sequential inductive bias. Instead of relying entirely on pairwise attention weights across all positions, the GRU updates the hidden state progressively, which helps smooth noisy signals in longer sequences.

The dual-gated design further extends the standard GRU by explicitly incorporating temporal interval and item frequency signals. Temporal gaps and interaction frequency influence user behavior in different ways. A long time interval may indicate preference drift, while high item frequency may reflect exposure bias or popularity effects. If these two signals are mixed implicitly, the model may not distinguish whether a change in behavior is due to time decay or repeated exposure. By assigning separate gates to temporal decay and frequency adjustment, the model can regulate their effects independently. This design makes the update process more controllable and easier to analyze, and it aligns with the practical observation that recency and popularity are distinct behavioral factors.

For short-term interest modeling, an attention mechanism is used instead of another recurrent layer. Short-term intent is often determined by a subset of recent interactions

rather than the entire sequence. Attention allows the model to dynamically assign higher weights to the most relevant recent behaviors. This is particularly useful when a user session contains mixed intents or when only a few recent actions are strongly related to the next item. Compared with purely recurrent modeling, attention provides more direct and flexible weighting over the recent window, which improves the representation of local intent.

Another important decision is to explicitly model temporal intervals and item frequency rather than relying only on positional encoding. Positional information captures order but does not reflect actual time gaps. Two interactions that are adjacent in position may be separated by minutes or by months, and their influence on preference evolution can be very different. Similarly, item frequency is not simply an item attribute; it interacts with user exposure patterns and may bias representation learning toward popular items. Treating time intervals and frequency as explicit input features allows the model to incorporate these behavioral signals directly and improves robustness across datasets with different temporal granularity and popularity distributions.

The clustering step before sequential modeling is mainly motivated by data sparsity and long-tail item distribution. In many real-world datasets, a large proportion of items have very few interactions. Learning reliable item-level representations under such sparsity is difficult. By grouping similar items into clusters, the model reduces categorical sparsity and captures higher-level semantic patterns. Although clustering reduces fine-grained specificity, it improves statistical efficiency and stabilizes long-term preference modeling, especially in sparse regimes.

In DiffLSRec, diffusion-based fusion is introduced to replace deterministic combination of long-term and short-term interests. Traditional fusion strategies, such as weighted averaging or gating, assume that the interaction between long-term preference and recent intent can be modeled by a single deterministic transformation. However, user behavior is often uncertain and may contain abrupt shifts or noisy interactions. The diffusion-based approach treats interest fusion as an iterative refinement process under stochastic perturbation. This allows the model to explore a richer representation space and better handle irregular behavioral patterns. Although diffusion increases computational cost and requires additional hyperparameter tuning, it provides greater flexibility in modeling uncertainty and complements the structured design of TiIfSRec.

Overall, the methodological choices in this thesis aim to balance structured modeling and flexibility. TiIfSRec emphasizes stable sequence modeling with explicit behavioral signals and practical efficiency. DiffLSRec extends this framework by incorporating probabilistic fusion to better capture uncertainty and complex interest interaction. Together,

these decisions reflect two different but complementary directions for modeling long-term and short-term interests in sequential recommendation.

TiIfSRec and DiffLSRec are designed to complement each other by adopting different fusion strategies for modeling long-term and short-term interests. TiIfSRec employs a structured and deterministic gating mechanism, which provides stable and efficient modeling of user behavior. In contrast, DiffLSRec enhances this framework by introducing a diffusion-based probabilistic fusion strategy, enabling more flexible and adaptive interaction between long-term and short-term representations through iterative refinement. In general, DiffLSRec tends to achieve better performance due to its ability to model complex interaction patterns and uncertainty. However, its effectiveness may be limited in scenarios where user behavior is highly sparse, noisy, or exhibits abrupt interest shifts, as reliable patterns may be difficult to learn. On the other hand, TiIfSRec is more suitable in settings with limited data, real-time requirements, or resource constraints, as its simpler architecture involves fewer parameters, offers better computational efficiency, and reduces the risk of overfitting. Therefore, DiffLSRec is preferred when sufficient data and computational resources are available and modeling flexibility is critical, while TiIfSRec is a more practical choice for efficiency-sensitive or data-constrained environments.

5.4 Research Value and Practical Implications

This section discusses the research value of TiIfSRec and DiffLSRec and their implications for practical recommendation systems. The focus is on clarifying what these models contribute beyond empirical performance gains and under which conditions they are most appropriate.

From a research perspective, TiIfSRec provides a structured approach to incorporating temporal intervals and item frequency into long- and short-term interest modeling. Instead of relying only on positional encoding or implicit attention mechanisms, the model explicitly separates temporal decay and frequency-aware adjustment through dedicated gating components. This design makes the influence of recency and exposure bias more transparent and easier to control. It also shows that introducing task-specific inductive bias can improve stability in sequential recommendation, especially in datasets with long-tail distributions and uneven interaction patterns.

DiffLSRec extends this line of work by modeling the interaction between long-term and short-term interests through a diffusion-based probabilistic process. Rather than combining these representations through a single deterministic transformation, the model treats fusion

as an iterative refinement procedure. This allows the representation to better accommodate uncertainty, irregular preference shifts, and noisy interactions. From a methodological perspective, this demonstrates that generative modeling techniques can be integrated into sequential recommendation frameworks to enhance flexibility in interest representation.

In practical systems, TiIfSRec is more suitable when computational efficiency and scalability are important. Its moderate parameter size and structured design make it easier to train and deploy in large-scale environments. The explicit modeling of time intervals and frequency information is particularly relevant in applications such as e-commerce or media platforms, where recency effects and popularity bias strongly influence user behavior.

DiffLSRec is more appropriate in settings where user behavior is highly dynamic or contains significant noise. In platforms with rapidly changing trends or diverse content types, the additional flexibility provided by probabilistic fusion may improve robustness. However, the higher computational cost and additional hyperparameters introduced by diffusion should be considered when system latency or resource constraints are strict.

The benefits of both models depend on certain data characteristics. Explicit time modeling requires reasonably reliable timestamp information. In extremely sparse datasets, clustering and frequency-aware modeling can help improve statistical efficiency, but may still be limited by insufficient interaction signals. Therefore, the selection between TiIfSRec and DiffLSRec should be guided by data quality, system constraints, and the desired balance between efficiency and modeling flexibility.

Overall, this thesis contributes two complementary approaches to long- and short-term interest modeling in sequential recommendation. TiIfSRec focuses on structured and efficient modeling with explicit behavioral signals, while DiffLSRec emphasizes flexibility in representing uncertainty during interest fusion. They provide alternative modeling options under different operational conditions rather than a single universal solution.

Chapter 6

Conclusion and Future Work

6.1 Introduction

This chapter concludes the thesis by bringing together the key ideas, findings, and insights developed throughout the work. The thesis focused on advancing sequential recommendation by addressing two central challenges: the difficulty of modeling long-term and short-term user interests in a flexible way, and the instability caused by noise, sparsity, and irregular behavioral patterns commonly found in real-world interactions. To address these issues, the thesis introduced two complementary modeling frameworks: a dual-gated time–frequency architecture for stabilizing and enriching sequential representations, and a diffusion-based generative refinement model for adaptive long–short interest fusion.

These contributions form a natural progression from structural sequence modeling to generative refinement. The dual-gated time–frequency model showed how temporal intervals and item frequency signals can help reduce popularity bias and strengthen low-frequency cues, leading to more balanced long-term representations. Building on this foundation, the diffusion-based framework viewed sequential recommendation as a multi-step denoising process, allowing the model to gradually incorporate short-term contextual information without overwhelming more stable long-term tendencies. Extensive experiments across multiple real-world domains confirmed the effectiveness of these approaches, demonstrating consistent gains in ranking performance, robustness under sparsity, and adaptability to rapid preference changes.

This chapter summarizes the main findings of the thesis and explaining their significance for sequential recommendation research. It then discusses the limitations of the proposed

models, focusing on practical concerns such as sparse interactions, computational cost, and irregular user behavior. Finally, the chapter outlines possible directions for future research, including opportunities to improve data efficiency, incorporate additional semantic or multimodal information, and extend generative refinement to broader recommendation scenarios. These discussions connect the thesis with the wider development of recommender systems and suggest meaningful way for further exploration.

6.2 Conclusion

This thesis presented two complementary sequential recommendation frameworks aimed at improving the modeling of long-term and short-term user interests under realistic conditions such as sparsity, noise, and irregular interaction patterns. The first framework, TiIfSRec, focused on stabilizing and enriching structural representations, while the second framework, DiffLSRec, introduced a generative refinement perspective for adaptive long–short interest fusion.

TiIfSRec addressed the challenge of uneven behavioral signals by jointly modeling time intervals and item frequency as two key factors influencing user preference evolution. To reduce sparsity and enhance representation stability, item categories were clustered using KMeans to generate compact item identifiers. In the long-term interest module, a GRU equipped with a time gate and a frequency gate allowed the model to regulate temporal decay and frequency-related adjustments more effectively. An attention mechanism was added to emphasize informative historical interactions, allowing the model to capture stable preference patterns over extended sequences. In the short-term interest module, GRU-based encoding incorporated time interval and positional information, which were combined with frequency signals and processed through self-attention. Experimental evaluations in three real-world recommendation settings demonstrated that TiIfSRec provides notable improvements across four standard ranking metrics, showing its ability to balance long-term stability with short-term specificity.

Building on the insights from TiIfSRec, the second framework, DiffLSRec, introduced a diffusion-based long–short interest fusion paradigm. Instead of relying on a fixed fusion structure or single-step aggregation, DiffLSRec refined user representations through a multi-step generative denoising process. At each diffusion step, short-term interaction cues guided the adjustment of the long-term preference state, enabling the model to incorporate new behavioral signals progressively without overwhelming stable historical tendencies. Token-level contextual enhancement and SNR-adaptive guidance further improved

the refinement process by strengthening relevant short-term signals while reducing the influence of noise or sparsity. Experiments conducted on the Cellphone, Clothes, and Movies datasets confirmed the effectiveness of this design, with DiffLSRec consistently outperforming traditional sequential recommenders and diffusion-based baselines across multiple evaluation measures.

TiIfSRec and DiffLSRec contribute a unified perspective on sequential recommendation. TiIfSRec improves the structural modeling of user behavior through time–frequency co-modeling and dual-gated GRUs, while DiffLSRec extends this representation into a multi-step refinement procedure capable of adapting to rapid preference changes. The combined findings highlight that sequential user modeling benefits not only from enriched temporal and frequency-aware signals, but also from viewing preference fusion as a gradual, context-guided process rather than a fixed, deterministic step.

Overall, this thesis demonstrates that integrating structural gating mechanisms with generative refinement offers a promising direction for developing more flexible, stable, and data-efficient sequential recommendation models. The insights and models lay a foundation for future research on adaptive user representation learning, particularly in environments characterized by sparse data, evolving preferences, and complex interaction dynamics.

6.3 Future Work

Although the proposed frameworks perform well across multiple datasets, there are still practical problems that future work could address. One of the most common issues is data sparsity. In many real applications, most users only interact with a few items, and many items appear very rarely in the logs. When the training data is too limited, the model struggles to learn reliable embeddings and tends to make unstable predictions. Future work could reduce this problem by generating additional training signals, for example by creating pseudo-interactions, expanding item features with simple semantic information, or using small generative modules to fill in missing behavioral patterns. These methods could help the model form more stable user and item representations in the early stages of training.

A related challenge is the cold-start problem for new users. Since both TiIfSRec and DiffLSRec rely on historical interaction sequences, their performance may be limited when little or no user history is available. In such cases, the models cannot reliably capture user preferences, leading to less accurate recommendations. Future work could address this issue by incorporating auxiliary information such as user profiles or contextual features to better initialize user representations and improve recommendation quality for new users.

Another direction is to make better use of information from related domains. Users often browse or purchase similar items across different categories, and similar items in different domains may share overlapping semantics. When one domain has limited data, it may be helpful to borrow information from another domain with richer interactions. Models could share part of the embedding space, or adapt a pretrained representation to the target domain. This approach would be especially useful for inactive users or long-tail items that do not appear often enough for the model to learn independently.

Richer contextual and multimodal information also offers a practical improvement path. Text descriptions, product images, structured attributes, and user reviews can provide valuable signals when interaction sequences are short or noisy. These features often contain information that does not appear in the interaction logs, such as product functionality, style, or user sentiment. Incorporating these features into the model could help represent rare items more accurately and support better predictions when user history is limited.

Another practical consideration is computational efficiency. DiffLSRec requires multiple refinement steps during inference, which increases processing time compared to single-step models. This is acceptable in offline evaluation, but may become an issue in real-time recommendation systems or large-scale platforms. Future work could explore faster approximate refinement procedures, reduce the number of denoising steps, or design partial-sampling mechanisms that keep most of the benefits of diffusion while reducing latency.

Overall, the methods proposed in this thesis show that combining structural time–frequency modeling with iterative refinement can improve sequential recommendation performance in realistic settings. Future developments that focus on better handling sparse data, transferring knowledge across domains, using additional contextual information, and improving inference efficiency may further enhance the practicality and scalability of such models in real-world applications.

References

[1] Dacheng Cai, Xiaofei He, Jiawei Han, and Thomas S. Huang. Graph regularized non-negative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1548–1560, 2011.

[2] Robin Devooght and Hugues 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.

[3] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 152–160, 2017.

[4] Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Jianfeng Qu, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor S. Sheng. Frequency enhanced hybrid attention network for sequential recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 78–88, 2023.

[5] Jiasheng Duan, Peng-Fei Zhang, Ruihong Qiu, and Zi Huang. Long short-term enhanced memory for sequential recommendation. *World Wide Web*, 26(2):561–583, 2022.

[6] Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *Proceedings of the 16th IEEE International Conference on Data Mining*, pages 191–200, 2016.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182, 2017.

[8] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 843–852, 2018.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, et al. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, pages 1–10, 2016.

[10] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *ArXiv*, abs/2403.03952, 2024.

[11] Yidan Hu, Yong Liu, Chunyan Miao, and Yuan Miao. Memory bank augmented long-tail sequential recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pages 791–801, 2022.

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272, Pisa, Italy, 2008.

[13] Zan Huang, Daniel Zeng, and Hsinchun Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5):68–78, 2007.

[14] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *Proceedings of the 18th IEEE International Conference on Data Mining*, pages 197–206, 2018.

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

[16] Hyunwoo Lee and Junsu Kim. Ediffurec: An enhanced diffusion model for sequential recommendation. *Mathematics*, 12(12):1795, 2024.

[17] Jiacheng Li, Yujie Wang, and Julian 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.

[18] Zehui Li, Aixin Sun, and Chenliang Li. Diffurec: A diffusion model for sequential recommendation. *ACM Transactions on Information Systems*, 42(3):66:1–66:28, 2024.

[19] Shangsong Liang, Zhou Pan, Wei Liu, Jian Yin, and Maarten de Rijke. A survey on variational autoencoders in recommender systems. *ACM Computing Surveys*, 56(10):268:1–268:40, 2024.

[20] Qiao Liu, Yifu Zeng, Zhenyao Zhou, Enhong Chen, and Hui Xiong. Stamp: Short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1831–1839, 2018.

[21] Siyi Liu and Yujia Zheng. Long-tail session-based recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 509–514, 2020.

[22] Yang Liu, Yitong Wang, and Chenyue Feng. UniRec: A dual enhancement of uniformity and frequency in sequential recommendations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1483–1492, 2024.

[23] Chen Ma, Peng Kang, and Xue Liu. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 825–833, 2019.

[24] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318, 2013.

[25] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2007.

[26] Qi Pi, Wei Bian, Guorui Zhou, Xiaoqiang Zhang, Xiaoqian Zhu, Kun Gai, and Yong Geng. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2671–2679, 2019.

[27] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys*, 51(4), 2018.

[28] Sandeep Kumar Rachamadugu, Jayanarayana Reddy Dwaram, and Kiran Rao Patike. Recommender system based on deep neural network and long short term memory. In *International Conference on Computing, Networking, Telecommunications and Engineering Sciences Applications*, pages 50–55, 2021.

[29] Anand Rajaraman and Jeffrey David Ullman. Data mining. In *Mining of Massive Datasets*, pages 1–17. Cambridge University Press, Cambridge, 2011.

[30] Steffen Rendle, Christoph Freudenthaler, and Lars 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.

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

[32] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 158–167, Minneapolis, MN, USA, 2000.

[33] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. Multi-rate deep learning for temporal recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 909–912, 2016.

[34] Jian Tang and Ke 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 369–377, 2018.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

[36] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 515–524, 2017.

[37] Shoujin Wang, Liang Hu, Yan Wang, et al. Sequential recommender systems: Challenges, progress and prospects. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 6332–6338, 2019.

[38] Wenjie Wang, Yuhui Xu, Fuli Feng, Xiangnan Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 832–841, 2023.

[39] Jiajin Wu, Bo Yang, Runze Mao, and Qing Li. Popularity-aware sequential recommendation with user desire. *Expert Systems with Applications*, 237, 2024.

[40] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 328–337, 2020.

[41] Weiqin Yang, Jiawei Chen, Shengjia Zhang, Peng Wu, Yuegang Sun, Yan Feng, Chun Chen, and Can Wang. Breaking the top-k barrier: Advancing top-k ranking metrics optimization in recommender systems. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3542–3552, 2025.

[42] Zhen Yang, Weitong Chen, and Jian Huang. Enhancing recommendation on extremely sparse data with blocks-coupled non-negative matrix factorization. *Neurocomputing*, 278:126–133, 2018.

[43] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. Multi-order attentive ranking model for sequential recommendation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 5709–5716, 2019.

[44] Liang Zhang, Lifang Peng, and Phelan C.A. Novel recommendation of user-based collaborative filtering. *Journal of Digital Information Management*, 12(3):165–175, 2014.

[45] Qian Zhao, F. Maxwell Harper, Gediminas Adomavicius, and Joseph A. Konstan. Explicit or implicit feedback? engagement or satisfaction? a field experiment on machine-learning-based recommender systems. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1331–1340, Pau, France, 2018.

[46] Yu Zhu, Hao Li, Yikang Liao, et al. What to do next: Modeling user behaviors by Time-LSTM. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3602–3608, 2017.

[47] Sharare Zolgahdr, Ole Winther, and Paul Jeha. Generative diffusion models for sequential recommendations. In *Proceedings of the 1st Workshop on Risks, Opportunities, and Evaluation of Generative Models in Recommender Systems*, 2024.

APPENDICES

This chapter provides the implementation details and Python code for the two sequential recommendation models developed in this thesis. The appendices ensure transparency and ease of replication by presenting the key steps of data preprocessing, training, and inference.

The organization of the appendices is as follows:

- **Appendix A** presents the main code for the TiIfSRec model, including the the long-term module and short-term module enhanced by time–frequency signals.
- **Appendix B** provides the key Python code for the DiffLSRec model, including the forward diffusion process and the reverse denoising procedure used in the diffusion-based fusion module.

These appendices provide direct access to the underlying implementation, making the model logic transparent and enabling straightforward adaptation in different experimental setups.

Appendix A

Python Code for TiIfSRec

This appendix provides code for the long-term interest module, the short-term interest module, which together form the core components of the TiIfSRec model. The data preprocessing procedures are also provided to support their operation in the TiIfSRec model.

A.1 Data Preprocessing

This section describes the data preprocessing procedures required for both the long-term interest module and the short-term interest module. The preprocessing steps include sequence construction, time interval calculation, frequency extraction, and item indexing, which prepare the raw interaction data into structured inputs suitable for model training. The same preprocessing pipeline is also reused by DiffLSRec to ensure consistent data formatting and comparable experimental conditions across models.

```
def preprocess_data(self):
    self.data = self.data.dropna(subset=['parent_asin', 'user_id', '
        timestamp', 'title'])
    self.data['timestamp'] = pd.to_datetime(self.data['timestamp'], errors=
        'coerce')
    self.data = self.data.dropna(subset=['timestamp'])
    self.data = self.data.sort_values(by=['user_id', 'timestamp'])
    self.generate_product_ids()
```

```
    # timestamp (seconds)
    self.data['time_seq'] = pd.to_datetime(self.data['timestamp']).astype(
        int) / 10 ** 9
    self.data['time_seq'] = self.data.groupby('user_id')['time_seq'].diff()
        .fillna(0).astype(float)
    mean_time_seq = self.data['time_seq'][self.data['time_seq'] > 0].mean()
    self.data['time_seq'] = self.data['time_seq'].apply(lambda x:
        mean_time_seq if x == 0 else x)

    # user history sequence
    self.data['product_sequence'] = self.data.groupby('user_id')['
        parent_asin'].transform(lambda x: x.tolist())

    # item frequency
    item_freq = self.data['product_id'].value_counts().to_dict()
    item_freq_inv = {key: 1 / value for key, value in item_freq.items()}
    self.data['item_freq'] = self.data['product_id'].map(item_freq_inv)

    # position
    self.data['position'] = self.data.groupby('user_id').cumcount()

    # scaler
    scaler = MinMaxScaler()
    self.data['time_seq'] = scaler.fit_transform(self.data[['time_seq']])
    self.data['item_freq'] = scaler.fit_transform(self.data[['item_freq']])

    self.preprocessed_data = self.data[
        ['user_id', 'product_sequence', 'product_id', 'time_seq', '
            item_freq', 'position']].drop_duplicates()


def pad_or_truncate(seq, max_len):
    if len(seq) >= max_len:
        return list(seq[-max_len:])
    else:
        return [0] * (max_len - len(seq)) + list(seq)

def data_for_model(path, max_seq_len):
```

```
if path.endswith(".csv"):
    data = pd.read_csv(path)
else:
    raise ValueError("Unsupported␣file␣format:␣must␣be␣.csv")
user_data = data.groupby('user_id')
item_seqs, time_seqs, freq_seqs, target_seqs = [], [], [], []
max_seq_index = max_seq_len - 1

for user_id, group in user_data:
    item_seq = group['product_id'].values
    time_seq = group['time_seq'].values
    freq_seq = group['item_freq'].values

    item_seq = pad_or_truncate(item_seq, max_seq_len)
    time_seq = pad_or_truncate(time_seq, max_seq_len)
    freq_seq = pad_or_truncate(freq_seq, max_seq_len)

    item_seqs.append(item_seq[:-1])
    time_seqs.append(time_seq[:-1])
    freq_seqs.append(freq_seq[:-1])
    target_seqs.append(item_seq[-1])

print("time_seq␣distribution:")
print(pd.Series(np.array(time_seqs).flatten()).describe())

print("freq_seq␣distribution:")
print(pd.Series(np.array(freq_seqs).flatten()).describe())

return (
    tf.constant(item_seqs, dtype=tf.int32),
    tf.constant(time_seqs, dtype=tf.float32),
    tf.constant(freq_seqs, dtype=tf.float32),
    tf.constant(target_seqs, dtype=tf.int32)
)
```

A.2 Code for Long-term Interest Module

This section describes the Python implementation of the long-term interest module, including the time–frequency enhanced GRU structure and the attention mechanism used to extract stable historical preferences.

```
item, time, freq = inputs

# Embedding layers
item_emb = self.embedding_item(item)
time_emb = self.time_dense(tf.expand_dims(time, -1))
freq_emb = self.freq_dense(tf.expand_dims(freq, -1))

batch_size = tf.shape(item_emb)[0]
seq_len = tf.shape(item_emb)[1]
h_prev = tf.zeros((batch_size, self.gru_units))

for t in tf.range(seq_len):
    item_seq = item_emb[:, t, :]
    time_seq = time_emb[:, t, :]
    freq_seq = freq_emb[:, t, :]

     # Time and frequency gates
    time_gate = tf.nn.sigmoid(
        tf.matmul(item_seq, self.W_xtg) +
        tf.matmul(time_seq, self.W_tg) +
        self.b_tg
    )

    freq_gate = tf.nn.sigmoid(
        tf.matmul(item_seq, self.W_xfg) +
        tf.matmul(freq_seq, self.W_fg) +
        self.b_fg
    )

    # Adaptive decay rate
    decay_in = tf.concat([time_gate, freq_gate], axis=-1)
```

```
delta_t = tf.nn.softplus(tf.matmul(decay_in, self.W_delta) + self.
    b_delta)
h_prev_decay = tf.exp(-delta_t) * h_prev # [B, H]

# Reset gate
reset_gate = tf.nn.sigmoid(
    tf.matmul(item_seq, self.W_xr) +
    tf.matmul(h_prev_decay, self.W_hr) +
    self.b_r
)

# Update gate
update_gate = tf.nn.sigmoid(
    tf.matmul(item_seq, self.W_xz) +
    tf.matmul(h_prev_decay, self.W_hz) +
    self.b_z
)

h_base = tf.nn.tanh(
    tf.matmul(item_seq, self.W_xh) +
    tf.matmul(reset_gate * h_prev_decay, self.W_hh) +
    self.b_h
)

# Frequency-direction candidate
h_freq = tf.nn.tanh(
    h_base +
    tf.matmul(freq_gate, self.W_f_dir) +
    self.b_f_dir
)

# Direction gate
psi = tf.nn.sigmoid(
    tf.matmul(freq_gate, self.W_psi) + self.b_psi
)

h_candidate = (1.0 - psi) * h_base + psi * h_freq
```

```
    # Attention
    concat_hx = tf.concat([h_prev_decay, item_seq], axis=-1)
    alpha_m = tf.nn.sigmoid(tf.matmul(concat_hx, self.W_a))
    h_att = alpha_m * h_prev_decay

    # Final hidden state update
    h_prev = (1.0 - update_gate) * h_att + update_gate * h_candidate

e_long = h_prev
```

A.3 Code for Short-term Interest Module

This section describes the Python implementation of the short-term interest module, which integrates time interval signals, positional information, and frequency features to model recent user behavior.

```
self.item_embedding = Embedding(input_dim=num_items, output_dim=embed_dim,
    mask_zero=True, embeddings_regularizer=tf.keras.regularizers.l2(0.01))
self.time_gru = GRU(embed_dim, return_sequences=True, name="TimeGRU")
self.freq_embedding = Embedding(input_dim=num_items, output_dim=embed_dim,
    mask_zero=False)
self.freq_dense = Dense(embed_dim, activation=None, name="FreqProjection")

self.WQ = Dense(embed_dim, name="QueryWeight")
self.WK = Dense(embed_dim, name="KeyWeight")
self.WV = Dense(embed_dim, name="ValueWeight")

self.ffn = Dense(embed_dim, activation="relu")
self.dropout = Dropout(0.5)
self.layer_norm = LayerNormalization(epsilon=1e-6)

self.output_layer = Dense(num_classes, activation='softmax', name="
   OutputLayer")
```

```
item_seq, time_seq, freq_seq = inputs

item_emb = self.item_embedding(item_seq)
time_emb = self.time_gru(tf.expand_dims(time_seq, -1))
freq_emb = self.freq_dense(tf.expand_dims(freq_seq, -1))

Q = self.WQ(item_emb)
K = self.WK(item_emb) + time_emb + freq_emb
V = self.WV(item_emb) + time_emb + freq_emb

dk = tf.cast(tf.shape(Q)[-1], tf.float32)
attention_logits = tf.matmul(Q, K, transpose_b=True) / tf.math.sqrt(dk)
attention_weights = tf.nn.softmax(attention_logits, axis=-1)
attention_output = tf.matmul(attention_weights, V)

ffn_output = self.ffn(attention_output)
ffn_output = self.dropout(ffn_output, training=training)
seq_output = self.layer_norm(attention_output + ffn_output)

e_short = tf.reduce_mean(seq_output, axis=1)
```

Appendix B

Python Code for DiffLSRec

This appendix provides code for the diffusion-based fusion module in DiffLSRec, including the forward diffusion step and the reverse denoising process. The implementation illustrates how noise is added to long-term representations, how short-term signals guide the refinement process, and how the model progressively reconstructs the fused embedding across multiple denoising iterations. The data preprocessing code used to prepare interaction sequences and extract temporal and frequency features is consistent with the preprocessing workflow described in Appendix A.

```
x_short, x_long = inputs

e_short, S_short_full = self.short_model(x_short, training=training,
   return_representation=True)
S_short = S_short_full[:, -self.k:, :]
e_long = self.long_model(x_long, training=training, return_representation=
   True)

B = tf.shape(e_short)[0]
d = self.embed_dim

# Forward
tT = self.T - 1
alpha_bar_T = tf.cast(self._alpha_bar[tT], e_short.dtype)
noise = tf.random.normal(tf.shape(e_long))
e_t = tf.sqrt(alpha_bar_T) * e_long + tf.sqrt(1.0 - alpha_bar_T) * noise
```

```
# Backward
for t in reversed(range(self.T)):
    t_idx = tf.fill([B], t)
    t_emb = self._timestep_embed(t_idx, d)
    t_emb = self.t_dense(t_emb)

    # Token-level Contextual Enhancement:
    c_t = self._token_context(e_t, S_short)

    gate_in = tf.concat([c_t, e_short, e_long], axis=-1)
    g = tf.sigmoid(self.Wg(gate_in))
    c_t = g * c_t + (1.0 - g) * e_short

    # Classifier-Free Guidance
    if training:
        mask = tf.cast(tf.random.uniform([B, 1]) > self.p_drop, e_t.dtype)
        c_cond = c_t * mask
    else:
        c_cond = c_t

    # eps
    eps_in_cond = tf.concat([e_t, t_emb, c_cond], axis=-1)
    eps_cond = self.eps_net(eps_in_cond)

    eps_in_uncond = tf.concat([e_t, t_emb, tf.zeros_like(c_t)], axis=-1)
    eps_uncond = self.eps_net(eps_in_uncond)

    # SNR
    s_t = self._snr_guidance_weight(t)
    eps_hat = eps_uncond + s_t * (eps_cond - eps_uncond)

    eps_small = tf.constant(1e-5, dtype=e_t.dtype)

    alpha_bar_t = tf.cast(self._alpha_bar[t], e_t.dtype)
    alpha_bar_t = tf.clip_by_value(alpha_bar_t, eps_small, 1.0 - eps_small)

    if t > 0:
```

```
            alpha_bar_tm1 = tf.cast(self._alpha_bar[t - 1], e_t.dtype)
            alpha_bar_tm1 = tf.clip_by_value(alpha_bar_tm1, eps_small, 1.0 -
                eps_small)
        else:
            alpha_bar_tm1 = tf.ones_like(alpha_bar_t)

        alpha_t = alpha_bar_t / alpha_bar_tm1
        alpha_t = tf.clip_by_value(alpha_t, eps_small, 1.0 - eps_small)
        beta_t = 1.0 - alpha_t

        x0_pred = (e_t - tf.sqrt(1.0 - alpha_bar_t) * eps_hat) / tf.sqrt(
            alpha_bar_t)

    def _token_context(self, e_t: tf.Tensor, S_short: tf.Tensor):
        Q = self.WQ(tf.expand_dims(e_t, axis=1))
        K = self.WK(S_short)
        V = self.WV(S_short)

        scale = tf.cast(tf.math.rsqrt(tf.cast(tf.shape(Q)[-1], tf.float32)), Q.
            dtype)
        attn = tf.matmul(Q, K, transpose_b=True) * scale
        attn = tf.nn.softmax(attn, axis=-1)
        ctx = tf.matmul(attn, V)

        return tf.squeeze(ctx, axis=1)


    def _snr_guidance_weight(self, t: int):
         u = 1.0 - (tf.cast(t, tf.float32) / tf.cast(tf.maximum(self.T - 1, 1),
             tf.float32))
        base = tf.sigmoid(6.0 * (u - 0.5))

        return self.smax * base
```