SCSE23-0406

TEMPORAL ATTENTION GRAPH-OPTIMIZED NETWORKS FOR SEQUENTIAL RECOMMENDATION

Submitted by: **Pathak Siddhant (U2023715K)**

Supervisor: **Assoc Prof. Ke Yiping, Kelly**

Examiner: **Dr. Huang Shell Ying**

School of Computer Science and Engineering

AY2023-24 S2

NANYANG TECHNOLOGICAL UNIVERSITY

SCSE23-0406

TEMPORAL ATTENTION GRAPH-OPTIMIZED NETWORKS FOR SEQUENTIAL RECOMMENDATION

Submitted in Partial Fulfilment of the Requirements
for the Degree of Bachelor of Engineering in Computer Science
of the Nanyang Technological University

Pathak Siddhant (U2023715K)

School of Computer Science and Engineering

AY2023-24 S2

Abstract

By analyzing a user's ongoing journey, sequential recommendation systems provide a more personalized and context-aware experience, enhancing user satisfaction and engagement. Sequential recommendation is crucial in various domains such as e-commerce, and social networking. Deep learning ameliorates and solves various research domains, including recommendation systems [1], [2], [3]. Deep learning models have become a robust alternative to traditional physical and statistical approaches, owing to their capacity for data mining and feature extraction. Recent studies have increasingly utilized Graph Neural Networks in the domain of deep learning. Graph-based approaches have gained popularity in recent years due to their effectiveness in modeling intricate user-item relationships. They excel in handling complex, interconnected data, enabling more accurate and context-aware recommendations. The goal of this study is to analyze the benefits of sequential recommendation from dynamic graphs that capture the dynamics of users' preference along with a temporal dimension and propose a robust architecture back-tested with rigorous experimental study.

Acknowledgments

I express my gratitude to Associate Professor Ke Yiping, Kelly and my PhD supervisor DiMarzio Bing Qingtian for their invaluable guidance, support and encouragement throughout the course of my research. Their insightful feedback and constructive criticism have been instrumental in shaping my research direction and improving the quality of my thesis.

I am grateful to the School of Computer Science and Engineering (SCSE) for offering me the opportunity to participate in high-quality research during my undergraduate studies. Additionally, I appreciate the support provided by the SCSE in terms of the essential hardware resources needed to conduct the various experiments for this Final Year Project.

Contents

Abstract **i**

Acknowledgments **ii**

Contents **iii**

List of Figures **vi**

List of Tables **viii**

1 Introduction **1**

1.1 Motivation 1

1.2 Research Gap 1

1.3 Objective 2

2 Literature Review **3**

2.1 Recommendation Systems 3

2.2 Sequential Recommendation 4

2.2.1 Top-K recommendation vs Top-K sequential recommendation 5

2.3 Deep learning frameworks 6

2.3.1 Graph Neural Networks 6

2.3.2 Transformers 7

2.3.3 Graph-based attention networks 8

3 Problem Statement **11**

3.1 Notations 11

3.2 Definitions 12

3.2.1 Continuous Time Bipartite Graph (CTBG) 12

3.2.2 Continuous Time Sequential Recommendation 13

4 Methodology **14**

4.1 Proposed Architecture 14

4.1.1 Embedding Module 14

4.1.2 Transformer Module 17

4.1.3 Prediction Module 21

4.2 Model Training and Optimization 22

5 Experimental Setup **24**

5.1 Datasets 24

5.2 Metrics 25

5.3 Evaluation Protocol 26

5.4 Parameter Settings 26

5.5 Baselines 27

5.5.1 Sequential Recommendation 27

5.5.2 Temporal Information 28

5.5.3 Graph-based recommendation 29

6 Results and Discussion **30**

6.1 Performance evaluation (RQ1) 32

6.2 Ablation Study 33

6.2.1 Transformer Layer (RQ2) 33

6.2.2 Encoding algorithm (RQ3) 35

6.2.3 Aggregation mechanism (RQ4) 38

6.3 Hyperparameter Sensitivity (RQ5) 40

6.3.1 Dimensionality of embeddings 41

6.3.2 Degree of neighbor sampling 43

7 Limitations and Future Works **46**

7.1 Explainability in Recommendations 46

7.2 Modeling long, short-term interests seperately 47

7.3 Temporal Embeddings Sampling Distribution 47

7.4 Use of Point-wise Convolutional Neural Networks 47

8 Conclusion **48**

Bibliography **53**

A Bochner Theorem **54**

B Training Graphs **56**

List of Figures

2.1 Architecture for Transformers (CS231: Deep Learning for Computer Vision, by Stanford University) 8
2.2 Architecture for RetaGNN, example of graph-based attention networks [32] 9
2.3 Architecture for TGSRec 10

3.1 An example of CTBG [29] 12
3.2 An example of user-item interaction sequence with collaborative sequential and temporal signals [29] 13

4.1 Schematic representation of our model's architecture 15

6.1 R@10 for five datasets with different number of layers 34
6.2 R@20 for five datasets with different number of layers 35
6.3 MRR for five datasets with different number of layers 35
6.4 R@10 for five datasets with different encoding algorithms 37
6.5 R@20 for five datasets with different encoding algorithms 37
6.6 MRR for five datasets with different encoding algorithms 38
6.7 R@10 for five datasets with different aggregation methods 39
6.8 R@20 for five datasets with different aggregation methods 40
6.9 MRR for five datasets with different aggregation methods 40
6.10 R@10 for five datasets with different embedding dimensions 42
6.11 R@20 for five datasets with different embedding dimensions 42
6.12 MRR for five datasets with different embedding dimensions 43
6.13 R@10 for five datasets with different degrees of neighbor sampling 44
6.14 R@20 for five datasets with different degrees of neighbor sampling 45
6.15 MRR for five datasets with different degrees of neighbor sampling 45

B.1 BPR Loss variation across Train set during training 56
B.2 Average Precision variation across Train and Validation sets during training.... 57
B.3 F1 Score variation across Train and Validation sets during training..... 57

List of Tables

3.1 Notations used and their descriptions 11

5.1 Dataset Statistics 25

5.2 Default parameter settings for experimentation 27

6.1 Performance Evaluation on Amazon Toys and Amazon Baby 31

6.2 Performance Evaluation on Amazon Music and Amazon Tools 31

6.3 Performance Evaluation on Movielens-100K 31

6.4 Ablation study on different number of transformer layers 34

6.5 Ablation study on different number of encoding algorithms 36

6.6 Ablation study on different number of aggregation mechanism 39

6.7 Dimensionality sensitivity analysis 41

6.8 Neighbor Sampling Degree sensitivity analysis 44

Chapter 1

Introduction

1.1 Motivation

In today's rapidly evolving digital landscape, recommendation systems have undergone a noteworthy transformation. They are experiencing a significant shift towards the implementation of sequential recommendation techniques. Such systems take into account the order of users' interactions and then predict their next action based on the context of what they have seen. By analyzing a user's ongoing journey, sequential recommendation systems provide a more personalized and context-aware experience, enhancing user satisfaction and engagement. Sequential recommendation is crucial in various domains such as e-commerce, and social networking [1].

1.2 Research Gap

Recent studies have increasingly utilized Graph Neural Networks in the domain of deep learning. Graph-based approaches have gained popularity in recent years due to their effectiveness in modeling intricate user-item relationships [4], [5], [6]. Graph Neural Networks have revolutionized recommendation systems by leveraging graph structures to capture intricate user-item interactions. They excel in handling complex, interconnected data, enabling more accurate and context-aware recommendations. However, the primary assumption made by them is to take into account only the interaction of items with users as

ordered sequences, neglecting the time component. For example, the demand for an item may be seasonal and might change in the future, or users' ratings may cause variations due to data distribution drifts, or the evolution of trends. By neglecting these temporal dynamics, traditional sequential recommendation models may not adapt adequately to real-world scenarios where time plays a crucial role in shaping user preferences and item popularity. There has been little research into the use of graph-based deep learning techniques for sequential and temporal recommendations.

1.3 Objective

The objective of this study is to contribute to the existing knowledge base on deep learning algorithms in the field of collaborative filtering and sequential recommendation. Previous research, due to its failure to keep longer context and requirement of substantial amounts of dense data structures, renders them infeasible for scalability purposes [7]. The goal of this study is to analyze the benefits of sequential recommendation from dynamic graphs that capture the dynamics of users' preference along with a temporal dimension and propose a robust architecture back-tested with a rigorous experimental study.

Chapter 2

Literature Review

This chapter systematically examines recommendation systems, focusing on sequential recommendation models and deep learning frameworks, including Graph Neural Networks, Transformers, and Graph-based attention networks.

2.1 Recommendation Systems

Recommendation systems have become pivotal in enhancing user experiences across various digital platforms. By predicting and suggesting items or services that users are likely to be interested in, these systems leverage sophisticated algorithms to make informed predictions. These predictions are primarily based on historical user behavior, employing mathematical models to navigate through vast datasets effectively.

The core of recommendation systems is built around mathematical models, notably collaborative filtering and content-based filtering [8]. These models scrutinize user interactions and item attributes to forecast user preferences and suggest pertinent items accordingly. There exist two principal methodologies within the realm of recommendation systems: neighborhood-learning and representation-learning.

Neighborhood models [9], [10] operate on the premise that users with similar preferences are likely to exhibit similar behavior. The cornerstone of this approach is the identification of similar users or items through the application of similarity metrics. This methodol-

ogy excels in uncovering highly localized relationships without necessitating any prior training phase. Utilizing the entire dataset, it seeks out subsets of similarities to generate recommendations.

In contrast, representation learning models [11], [12], [13], [14] focus on deriving low-dimensional representations from user-item interactions. This approach not only considers the direct interactions but also incorporates contextual information such as timestamps, time-delta, and demographic details to enrich the recommendation process. By training predictive models to learn latent features, these frameworks achieve a deeper understanding of user preferences and item characteristics. The fundamental objective of recommendation systems is encapsulated by the following mathematical formulation:

$$\text{argmax}_{i \in I} \mathbb{P}(i|(u, S))$$

Here, given a user u and their interaction sequence $S = [i_1, i_2..i_n]$ with items i, the system aims to maximize the probability of recommending an item i that aligns with the user's historical behavior.

Recommendation systems play a crucial role in modern digital ecosystems, guiding users through a personalized digital landscape [3]. Through the application of neighborhood-learning and representation-learning models, these systems uncover patterns and preferences, thereby enhancing the overall user experience.

2.2 Sequential Recommendation

Sequential Recommendation (SR) systems represent a significant advancement in the field of recommendation systems, shifting the focus towards understanding and predicting user preferences based on the temporal sequence of interactions. Unlike traditional Recommendation Systems (RS) that aim to map out a user's overall preferences, SR systems delve into the sequential patterns of user-item interactions, emphasizing the importance of the order and timing of these interactions [15], [6].

SR systems are predicated on the observation that a user's recent interactions with items

can provide valuable insights into their immediate interests and likely future actions. This approach to recommendation seeks to model both the long-term preferences and the short-term intents of users, thereby offering predictions that are not only relevant but also timely.

Given a user u and their sequence of interactions with items $S = [(i_1, t_1), (i_2, t_2)..(i_n, t_n)]$, where each interaction is represented as a tuple of the item i and the time t of interaction, SR systems aim to maximize the probability $P(i|(u, S))$ of recommending the next item i that the user is most likely to engage with.

$$\text{argmax}_{i \in I} \mathbb{P}(i|(u, S))$$

Sequential Recommendation systems represent a nuanced and highly effective approach to enhancing the user experience through personalized recommendations [16], [17]. By leveraging the sequential order and temporal dynamics of user-item interactions, these systems offer a sophisticated means of understanding and catering to the evolving preferences of users. The integration of session-based methodologies within the broader framework of SR systems highlights the ongoing innovation in the field, aiming to refine the precision and relevance of recommendations in an increasingly digital world.

2.2.1 Top-K recommendation vs Top-K sequential recommendation

Top-N recommendation systems have emerged as a critical component in the landscape of personalized recommendation engines, aiming to curate a set of N items from a vast collection that best matches the user's preferences. This methodology is widely applied across diverse domains, ranging from academic literature to music streaming and online education, facilitating enhanced user engagement through tailored recommendations.

Contrary to traditional Top-N recommendation systems, Top-N sequential recommendation models incorporate a critical dimension to the recommendation process: the sequence of user-item interactions. This approach recognizes the significance of the order in which items are engaged with by users, positing that this sequence profoundly influences future

interactions.

Top-N sequential recommendation systems model each user as a timeline of past item interactions, aiming to predict the next set of N items that a user is likely to engage with. The methodology underscores the temporal dynamics of user preferences, acknowledging that the relevance of recommendations can be greatly enhanced by considering the chronological context of past interactions.

$$\text{Objective} : \max \mathbb{P}(\text{Top-N items}|\text{User sequence})$$

2.3 Deep learning frameworks

2.3.1 Graph Neural Networks

Graph Neural Networks (GNNs) are designed to capture the intricate structures and relationships within graph-based data, effectively modeling the connections between nodes (representing entities such as users and items in recommendation systems) [18], [19], [20], [21]. The core mechanism of GNNs lies in their ability to propagate information through the graph via a process known as message passing. During each iteration or layer, a node's representation is updated based on the aggregated information from its neighbors, allowing the network to learn complex patterns across the graph. The update rule for a node v at layer $l + 1$ can be formalized as follows:

$$h_v^{(l+1)} = \text{UPDATE}^{(l)} \left(h_v^{(l)}, \text{AGGREGATE}^{(l)} \left(\{ h_u^{(l)} : u \in \mathcal{N}(v) \} \right) \right)$$

Here, $h_v^{(l)}$ denotes the feature representation of node v at layer l, $\mathcal{N}(v)$ represents the set of neighbors of v, and the functions $\text{UPDATE}^{(l)}$ and $\text{AGGREGATE}^{(l)}$ denote the update and aggregation operations at layer l, respectively. The aggregation function collects features from a node's neighbors, and the update function integrates the aggregated information with the node's current features to produce a new representation. This iterative process allows GNNs to effectively capture both local structures and global graph properties,

enabling deep insights into the data's relational context.

Sequential recommendation systems greatly benefit from the incorporation of GNNs [15], [22], [23], [17], [24], particularly due to their capability to model dynamic interactions over time. Through constructing graphs that embody the sequential nature of user-item interactions, GNNs adeptly unravel temporal patterns and dependencies. This temporal awareness enables the prediction of how user preferences evolve, significantly improving the accuracy of future interaction forecasts. GNN-based sequential recommendation systems excel in delivering timely and contextually relevant suggestions, thus elevating the user experience in ever-changing environments.

2.3.2 Transformers

Transformers [25] revolutionize sequence modeling by relying entirely on masked multi-attention mechanisms to process input data vectors, eschewing the recurrent layers commonly found in previous architectures. This design enables the model to analyse and weigh the importance of different components of the input data independently of their sequential order, allowing for parallel computation and significantly improving efficiency. The core idea is to generate attention scores which dictate the influence of each part of the input sequence on others.

The Transformer architecture consists of an encoder and a decoder, each comprising multiple layers that perform self-attention and position-wise feedforward operations. For any given position i in the input sequence, the self-attention mechanism computes a weighted sum of all positions' embeddings to produce an output that reflects the contributions of each position:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Here, Q, K, and V represent matrices for the queries, keys, and values, respectively, derived from the input embeddings, and d_k is the dimension of the key vectors, which is used to scale the dot products to prevent extremely large values. The softmax operation is applied to the result of the scaled dot product, yielding the weights for the values in V.

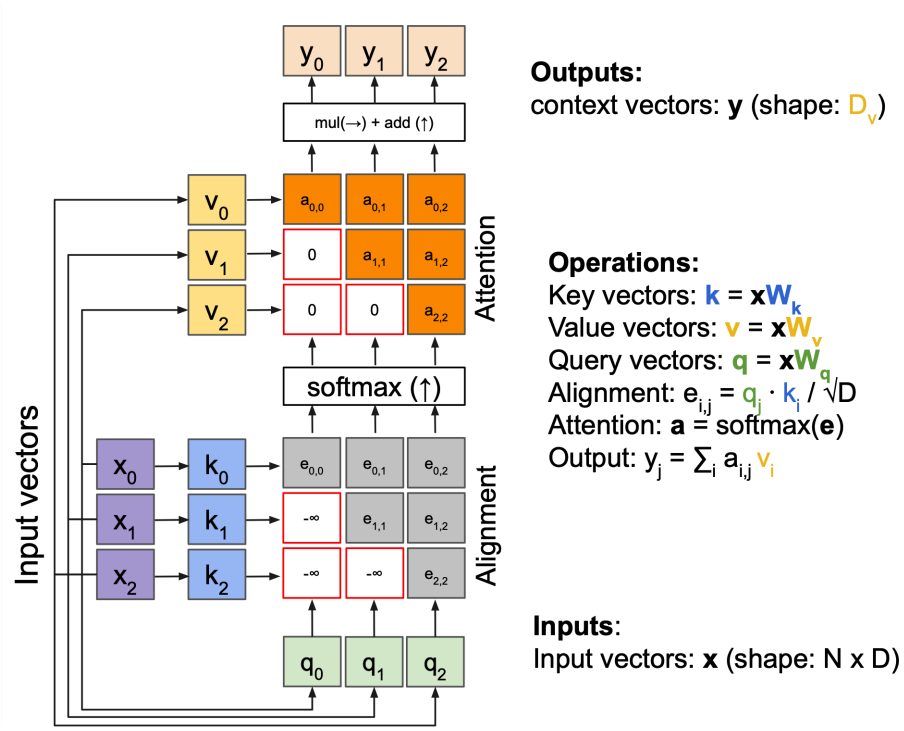


Figure 2.1: Architecture for Transformers (CS231: Deep Learning for Computer Vision, by Stanford University)

Each layer in both the encoder and decoder follows with position-wise feedforward networks, which apply two linear transformations with a ReLU activation in between:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

This architecture, with its reliance on self-attention and position-wise feedforward networks, allows Transformers to capture complex dependencies and understand the relative positioning of elements within the sequence, without being constrained by the sequence's length or the need for sequential data processing [26], [27], [28].

2.3.3 Graph-based attention networks

Graph-based attention networks, a variant of Graph Neural Networks (GNNs), introduce an attention mechanism to weigh the significance of nodes' contributions in the feature aggregation process [29], [30], [31], [32].

This approach enables the model to focus on more relevant information from neighboring

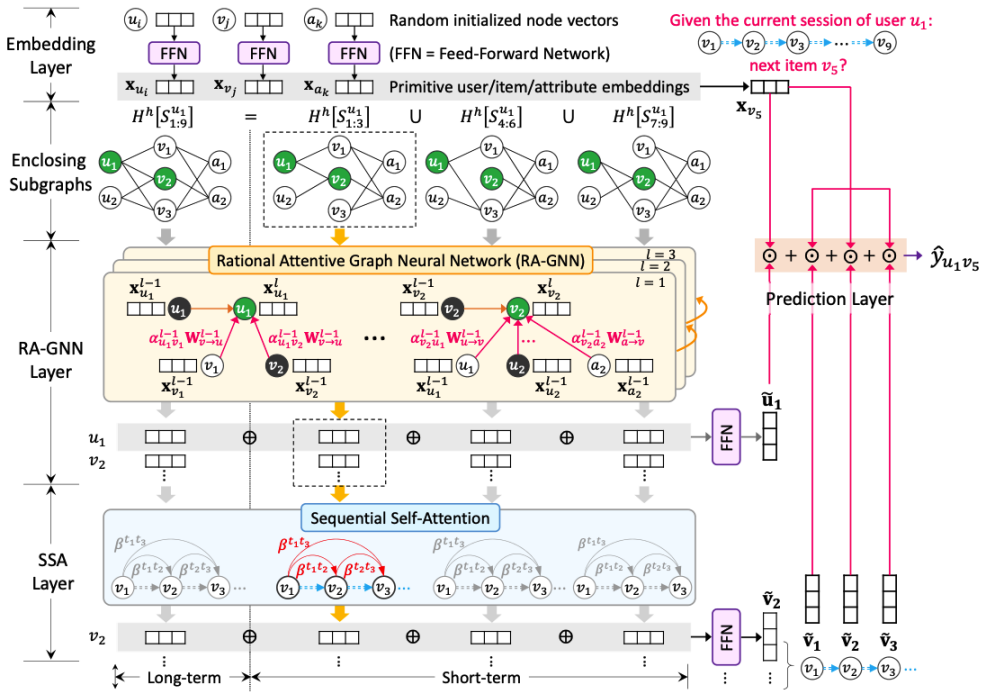


Figure 2.2: Architecture for RetaGNN, example of graph-based attention networks [32]

nodes, enhancing its ability to capture nuanced relational patterns within the graph. The attention mechanism computes attention coefficients that reflect the importance of each neighbor's features to a node, allowing for a dynamic and adaptive aggregation process. For a node v, the attention coefficient between v and its neighbor u in layer l can be expressed as:

$$\alpha_{vu}^{(l)} = \text{softmax}\left(\text{ATTENTION}^{(l)}\left(h_v^{(l)}, h_u^{(l)}\right)\right)$$
$$\text{ATTENTION}^{(l)}\left(h_v^{(l)}, h_u^{(l)}\right) = \text{LeakyReLU}\left(a^{(l)T}\left[W^{(l)}h_v^{(l)} \| W^{(l)}h_u^{(l)}\right]\right)$$

Here, $W^{(l)}$ is a weight matrix associated with layer l, $a^{(l)}$ is a learnable parameter vector for the attention mechanism, and $\|$ denotes concatenation. The attention coefficients $\alpha_{vu}^{(l)}$ are normalized across all neighbors k of node v using the softmax function to ensure they sum to 1, allowing them to be interpreted as probabilities that indicate the importance of each neighbor's contribution to the next layer's node representation. The updated feature

representation for node v at layer $l+1$ then incorporates these attention coefficients:

$$h_v^{(l+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(l)} W^{(l)} h_u^{(l)}\right)$$

In this equation, σ denotes an activation function, such as ReLU. Through this attention-based feature aggregation, graph-based attention networks can adaptively learn which edges (relationships) are more significant, thereby achieving a more discriminative representation of the nodes based on the graph's structural and feature-related information.

Temporal Graph Sequential Recommender

The authors introduce the Temporal Graph Sequential Recommender (TGSRec) [29], a novel framework for sequential recommendation that addresses the evolution of user preferences over time.

Central to TGSRec is the introduction of the Temporal Collaborative Transformer (TCT) layer, which enhances the traditional self-attention mechanism by incorporating collaborative attention, enabling the model to effectively capture collaborative signals across users and items while accounting for temporal dynamics. The TCT layer employs a collaborative attention mechanism where the query input comes from the target node (user/item), and the key and value inputs are derived from connected neighbors, integrating temporal information to evaluate the significance of interactions. This allows TGSRec to capture the evolving nature of user preferences and item relevancy over time, addressing the challenge of simultaneously encoding sequential patterns and collaborative signals.

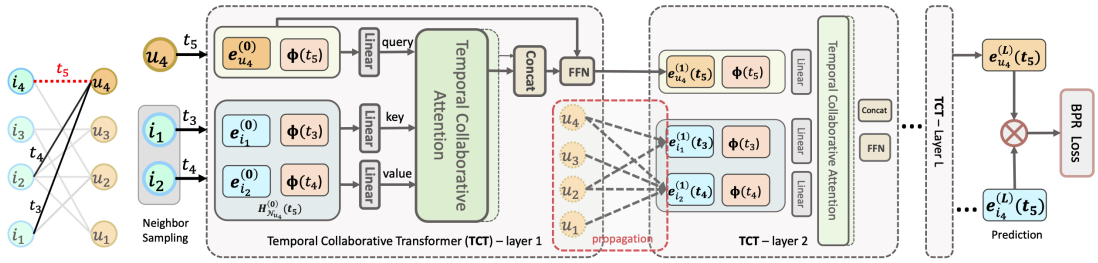


Figure 2.3: Architecture for TGSRec

Chapter 3

Problem Statement

3.1 Notations

Given a set of users U and a set of items I, with $V = U \cup I$ representing the joint user-item set, and a set of interactions $\{u, i, t\}$ where a user u interacts with an item i at a timestamp t, the goal of a sequential recommendation system is to predict the next item that a user is likely to interact with.

Each user u and item i is related to the other numerically with a learnt embedding $e_u^l(t)$ and $e_i^l(t)$, respectively, which captures the preferences and characteristics at the lth-layer of a multi-layered representation. The embeddings are temporal, implying they can change with time t, reflecting the evolving preferences of users and properties of items. The sequential recommendation system aims to leverage these embeddings and historical

Table 3.1: Notations used and their descriptions

Notation	Description
$u \in U$	User u from the user set
$i \in I$	Item i from the item set
$V = U \cup I$	Joint user-item set
$t \in T$	Timestamp t from the interaction
$\{u, i, t\}$	Triplet of user u, item i, timestamp t
$\lvert I \rvert$	Number of items in item set
$\lvert U \rvert$	Number of users in user set
$e_u^l(t) \in \mathbb{R}^d$	User learnt embedding at l^{th}-layer
$e_i^l(t) \in \mathbb{R}^d$	Item learnt embedding at l^{th}-layer
$E = [E_U; E_I] \in \mathbb{R}^{d \times \lvert V \rvert}$	Embedding table

interaction patterns to forecast future interactions.

The system learns from an embedding table $E = [E^U; E^I] \in \mathbb{R}^{d \times |V|}$, where $|V|$ is the cardinality of the joint set and d is the dimensionality of the embedding space, to make these predictions.

The recommendation model must effectively utilize the sequential nature of the user-item interactions, taking into account the order in which interactions occur and how user preferences evolve over time to make accurate predictions.

3.2 Definitions

3.2.1 Continuous Time Bipartite Graph (CTBG)

We introduce a bipartite graph in continuous time for recommendation purposes, which comprises N nodes and E edges, represented as $\mathcal{B} = \{\mathcal{U}, \mathcal{I}, \mathcal{E}_T\}$. In this graph, $\mathcal{U}$ and $\mathcal{I}$ constitute two separate sets of nodes corresponding to users and items, respectively. Each edge within $\mathcal{E}_T$ is characterized by a triplet $e = (u, i, t)$, with u being a member of $\mathcal{U}$, i belonging to $\mathcal{I}$, and t representing the time of interaction in $\mathbb{R}^+$. The triplet (u, i, t) signifies the engagement of user u with an item i at the specific timestamp t.

The subject of this study is the Sequential Recommendation (SR) issue within the realm of continuous-time events. Let $I_u(t)$ signify the collection of items that user u has interacted with up to the timestamp t, and $\mathcal{I} \setminus I_u(t)$ represent the set of items not yet interacted with. The continuous-time sequential recommendation challenge addressed in this paper is thus delineated.

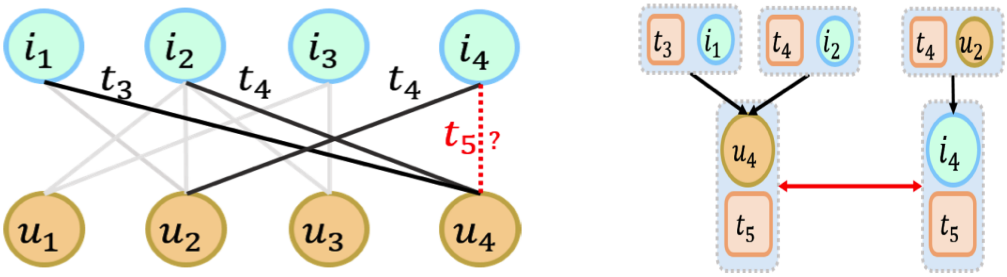


Figure 3.1: An example of CTBG [29]

3.2.2 Continuous Time Sequential Recommendation

For a particular user u, provided with a series of future timestamps $T_u > T$, the task is to offer a continuous-time sequential recommendation at each timestamp $t \in T_u$.

Our method uniquely accounts for timestamps as opposed to merely focusing on sequence order or position. This distinction means that, unlike previous approaches that assess sequential recommendations through next-item prediction, our approach requires the inclusion of future timestamps for making predictions. Should timestamps merely serve as sequence positions, our problem simplifies to the traditional model that relies solely on order or position data. It's crucial to recognize that timestamps may represent any real number, thereby facilitating a continuous model.

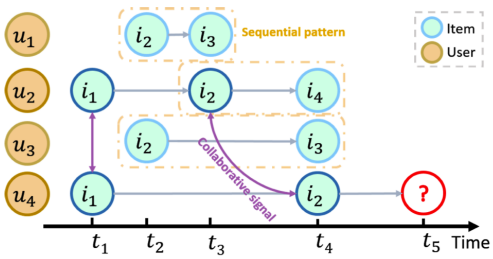


Figure 3.2: An example of user-item interaction sequence with collaborative sequential and temporal signals [29]

Chapter 4

Methodology

4.1 Proposed Architecture

Our proposed architecture consists of three principal components: the process begins with embedding learning and concludes with a probabilistic prediction for the SR problem.

The model processes graph-structured data through a composite suite. The architecture is intentionally designed to ensure seamless information transfer for advanced interaction modeling. The *Embedding Module* begins with node representation, transforming temporal and sequential data into numerical embeddings. The *Attention Module* leverages dual attention strategies to enhance the embeddings and to decipher user-item interaction patterns, deploying a cross-attention mechanism for critical sequential feature extraction, and a self-attention mechanism. The final segment, the *Outcome Module*, integrates the insights to predict user-item interaction scores, capturing the dynamism of interactions.

4.1.1 Embedding Module

The embeddings for users and their temporal neighbors (the items with which they have engaged) are determined through a bifurcated approach to parse their sequential and temporal attributes, specifically the persistent embeddings of nodes, and the dynamic temporal embeddings.

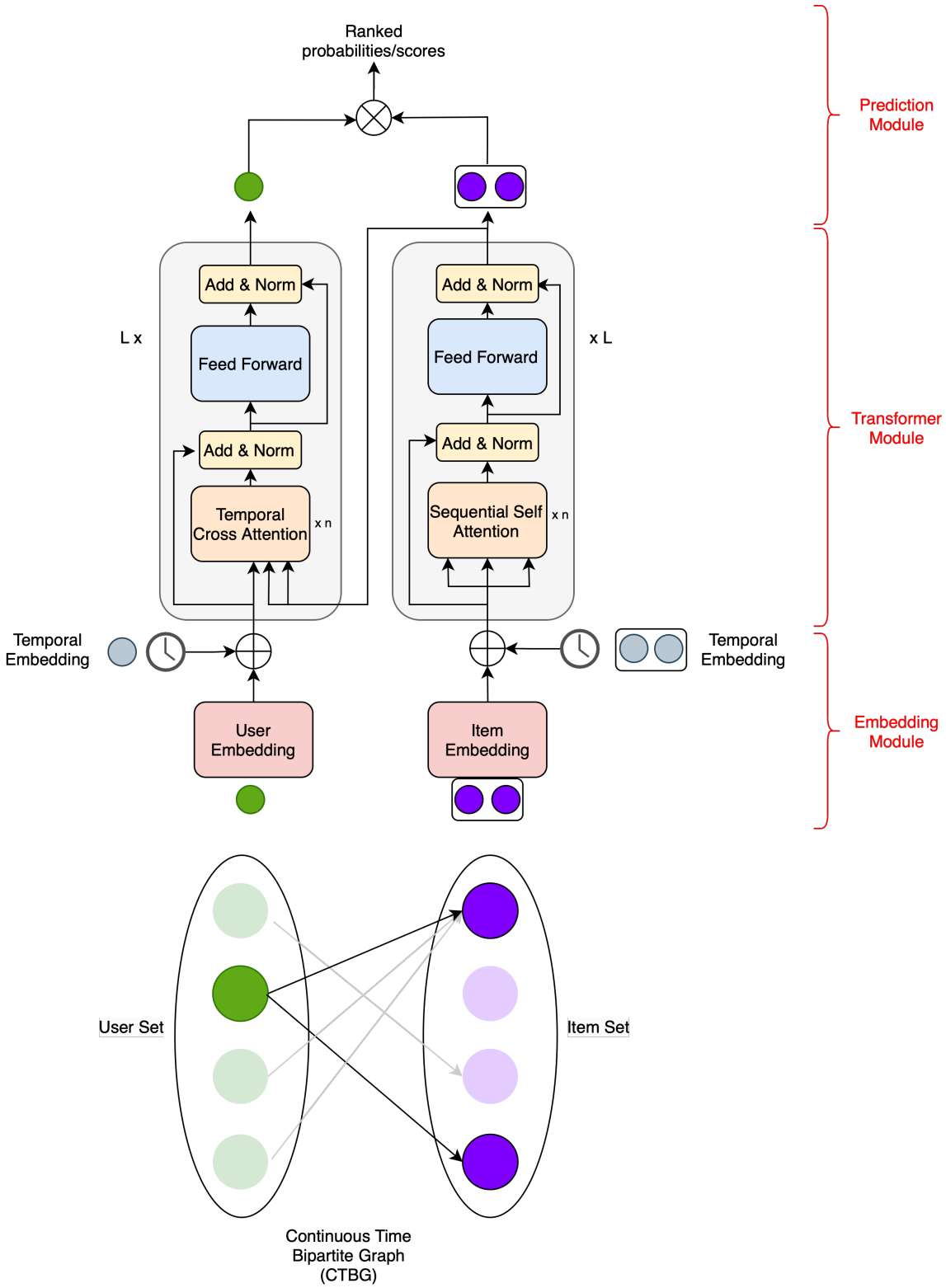


Figure 4.1: Schematic representation of our model's architecture

Feature embedding formulation

Experimental evidence suggests that persistent embeddings, which encode historical user-item interactions, are valuable in the domain of SR. We initialize user embeddings as e_u (likewise, e_i for items). These embeddings will be refined during the training process.

Temporal embedding formulation

The time span (or the relative interval) between interactions is crucial for capturing the dynamics of the temporal domain, revealing trends in content preferences and shifts in user interest. We introduce a time-encoding function module that transposes timestamps into corresponding vectors. For a user u and interactions i_1, t_1 and i_2, t_2, the **dynamic impact** is postulated as a function of the elapsed time between these events. Formally, it is expressed as:

$$\psi(t_1 - t_2) = \kappa(t_1, t_2) = \phi(t_1) \odot \phi(t_2) \tag{4.1}$$

The dynamic impact $\psi(t_1 - t_2)$ quantifies the temporal linkage between two interactions. Further, the time-encoding function is designed to be extensible to novel timestamps, allowing the model to deduce time spans not observed during training. Rather than just considering absolute time differences, positing dynamic impacts as a kernel facilitates generalization across any timestamp, modeling the temporal representations directly. Hence, the dynamic impact of any two timestamps can be deduced through the scalar product of their time representations. Equation 4.1 is realized by a continuous and shift-invariant kernel $\kappa(t_1, t_2)$, grounded in **Bochner's Theorem** [33]. The dynamic embedding is therefore articulated as:

$$\phi(t) \mapsto \sqrt{\frac{1}{\delta_T}}[\cos(\theta_1 t), \sin(\theta_1 t), \ldots, \cos(\theta_{\delta_T} t), \sin(\theta_{\delta_T} t)]^\intercal \tag{4.2}$$

where $\theta = [\theta_1, \theta_2, \ldots, \theta_{\delta_T}]^\intercal$ represents a set of tunable parameters, with δ_T specifying the dimensionality.

Please refer to Appendix A for a detailed explanation on Bochner's Theorem from classical harmonic analysis.

4.1.2 Transformer Module

This segment illuminates the innovation embodied in the Temporal Attention layer of our architecture. It assimilates the long-term and temporal embeddings generated previously, harnessing attention mechanisms to discern sequential and collaborative patterns through the utilization of both cross and self-attention mechanisms. The intricacies of these components are further expounded in the ensuing subsections, drawing inspiration from the seminal work "Attention is all you need" to construct our elaborate architecture.

Input Formulation

The aim of this submodule is to prepare the acquired embeddings for synergistic analysis. For the lth level of the attention layer, the *query* constitutes a merged form of user embeddings in concert with their temporal markers. For a user u at an instance t, the *query* is expressed as:

$$h_u^{(l)}(t) = e_u^{(l)}(t) \oplus \phi(t) \tag{4.3}$$

where l signifies the layer index, $h_u^{(l)}(t) \in \mathbb{R}^{f+f_T}$ represents the synthesized user information at time t, $\alpha_u^{(l)}(t) \in \mathbb{R}^f$ is the user's temporal representation, and $\Theta(t)$ captures the time-specific encoding. The symbol $\oplus$ indicates the concatenation process. For layers beyond the initial, $v_u^{(l)}(t)$ integrates insights from preceding layers, analogous to the message-passing scheme of graph neural networks.

Furthermore, to incorporate the temporal collaborative context of neighboring interactions, we select S distinct prior interactions of u leading up to time t denoted as $\mathcal{N}_u(t) = \{(i, t_p) \mid (u, i, t_p) \in \mathcal{E}_t \wedge t_p < t\}$.

Parallel to the user-based framework, item-centric information at t_s is analogously constituted, as illustrated in 4.3. Taking an item i at time t_s, we characterize the item's embedding thus:

$$h_i^{(l)}(t_s) = e_i^{(l)}(t_s) \oplus \phi(t_s) \tag{4.4}$$

Propagation and Accumulation in Neighborhood

Following the construction of inputs, we proceed to disseminate the assembled data from the neighbors denoted by $N_u(t)$ to deduce the dynamic embeddings. With the neighbors' involvement at time t, we integrate sequential patterns with collaborative temporal signals. The linear aggregate of the data from all sampled interactions is computed as follows:

$$e_{N_u}^{(l)}(t) = \sum_{(i,t_s)\in N_u(t)} \pi_t^u(i,t_s) W_v^{(l)} h_i^{(l-1)}(t_s) \tag{4.5}$$

where $\pi_t^u(i,t_s)$ signifies the relevance of an interaction $\{u,i,t_s\}$, and $\Lambda_v \in \mathbb{R}^{\delta_p \times (\delta_p+\delta_Q)}$ is a transformation matrix. The term $\pi_t^u(i,t_s)$ quantifies the historical interaction's $\{u,i,t_s\}$ influence on the user and item embedding at time t, a concept elaborated upon in the subsequent attention-centric sub-layers. This methodology resonates with the message-passing schema proposed in Graph Convolutional Networks.

Sequential Self Attention

Intrinsic sequence-attention mechanisms, when fused with dynamic embeddings, adeptly map out the evolving significance and influence of entities over time. The model is endowed with the capacity to assess historical interactions by their temporal relevance. This methodology guarantees a comprehensive grasp of both the sequence and the changing relevance of entities, which augments prediction accuracy for use cases like recommender systems and temporal sequence analysis. The integration of dynamic embeddings empowers the model to detect and accommodate evolving temporal patterns, bolstering its predictive proficiency.

A pioneering intrinsic sequence-attention schema is utilized to calculate the weights $\pi_t^i(i,t_s)$, taking into account intra-sequence interactions and corresponding temporal data. These aspects jointly inform the historical significance of interactions. The attention weight $\pi_t^i(i,t_s)$ is delineated as:

$$\pi_t^u(i,t_s) = \frac{1}{\sqrt{\delta_p+\delta_Q}} \left(W_k^{(l)} h_i^{(l-1)}(t_s)\right)^T \cdot W_q^{(l)} h_i^{(l-1)}(t), \tag{4.6}$$

where $W_k^{(l)}$ and $W_q^{(l)}$ signify linear transformation matrices. The coefficient $\frac{1}{\sqrt{\delta_p+\delta_Q}}$ ensures the dot product's magnitude remains controlled within high-dimensional spaces. Dot-product attention is preferred, as it simplifies the representation of the right-hand side, in the absence of transformation matrices and the normalizing factor, as:

$$e_i^{(l-1)}(t) \odot e_i^{(l-1)}(t_s) + \phi(t) \odot \phi(t_s), \tag{4.7}$$

with the first term representing the item-item sequential signal and the second term encapsulating the temporal influence as per Eq. 4.2. As layers accumulate, collaborative and temporal signals intertwine, endowing dot-product attention with the capacity to encapsulate temporal dynamics. Subsequently, attention weights are normalized across sampled interactions through a softmax function:

$$\pi_t^i(i, t_s) = \text{Softmax}(\pi_t^i(i, t_s))$$

Furthermore, computational efficiency is achieved by aggregating the data of all sampled interactions into a matrix $H^{(l-1)} \in \mathbb{R}^{(\delta_p+\delta_Q)\times S}$. We define

$$\begin{aligned} K_i^{(l-1)}(t) &= W_k^{(l)} H_{N_i}^{(l-1)}(t) \\ Q_i^{(l-1)}(t) &= W_q^{(l)} h_i^{(l-1)}(t), \text{and} \\ V_i^{(l-1)}(t) &= W_q^{(l)} h_i^{(l-1)}(t) \end{aligned}$$

as the respective key, value, and query inputs for the attention mechanism.

For clarity and to avoid complexity, we omit the layer superscripts and the temporal index t, amalgamating the two equations. We can rephrase the expression as:

$$e_{\mathcal{N}_i(t)} = V_i \cdot \text{Softmax}\left(\frac{K_i^T q_i}{\sqrt{\delta_p + \delta_Q}}\right), \tag{4.8}$$

thus mirroring the structure of dot-product attention in the Transformer model. The incorporation of multi-head attention permits the combination of outputs from each head, amalgamating information for subsequent aggregation. The justification for deploying

dot-product attention lies in its efficacy in discerning similarities across high-dimensional datasets. By deconstructing the attention framework to include interactions and temporal dynamics, our model is adeptly positioned to disentangle and harness these signals, substantially improving the accuracy of predictions.

Temporal cross-attention

The cross-user item attention mechanism is designed to assess the relevance of items to users by considering not only a user's historical interactions but also how similar users have interacted with these items. By analyzing these interactions in aggregate, the model can identify and leverage patterns that might be obscured when examining user-item interactions in isolation. This approach enables the model to make more informed predictions, enhancing its ability to recommend items that are likely to be of interest to the user. By analyzing these interactions in aggregate, the model can identify and leverage patterns that might be obscured when examining user-item interactions in isolation.

In alignment with the notations utilized in preceding sections, we introduce a cutting-edge cross attention mechanism for calculating the attention weights $\pi_t^u(i, t_s)$. This mechanism accounts for both the interactions between users and items and their associated temporal dynamics. The calculation of the attention weight $\pi_t^u(i, t_s)$ is presented as:

$$\pi_t^u(i, t_s) = \frac{1}{\sqrt{\delta + \delta_T}} \left(W_k^{(l)} h_i^{(l-1)}(t_s) \right)^T \cdot W_q^{(l)} h_u^{(l-1)}(t), \tag{4.9}$$

where the matrices $W_k^{(l)}$ and $W_q^{(l)}$ represent linear transformations, and the term $\frac{1}{\sqrt{\delta+\delta_T}}$ is employed to prevent the magnitude of the dot product from becoming excessively large in higher dimensions. The rationale for utilizing dot-product attention is elucidated by simplifying the right-hand side, in the absence of the transformation matrices and scaling factor, as:

$$e_u^{(l-1)}(t) \cdot e_i^{(l-1)}(t_s) + \Phi(t) \cdot \Phi(t_s), \tag{4.10}$$

indicating the sequential correlation between items and the temporal influence. Through the application of additional layers, the interplay between collaborative and temporal signals becomes increasingly intertwined, highlighting the dot-product attention's capa-

bility to capture temporal influences effectively. Subsequently, attention weights for all considered interactions are normalized using a softmax function:

$$\pi_t^u(i, t_s) = \text{Softmax}(\pi_t^u(i, t_s))$$

We define $K_u^{(l-1)}(t), V_i^{(l-1)}(t), Q_i^{(l-1)}(t)$ as the key, value, and query inputs for the attention mechanism, respectively, in a similar fashion, and thus simplifying the equation to:

$$e_{\mathcal{N}_u(t)} = V_i \cdot \text{Softmax}\left(\frac{K_i^T q_i}{\sqrt{\delta + \delta_T}}\right), \tag{4.11}$$

aligning with the structure of dot-product attention within the Transformer architecture. This alignment permits the implementation of the multi-head attention mechanism, where outputs from individual heads are concatenated to aggregate information.

Information Accumulation

To derive the temporal node embedding, the terminal operation within a transformer layer entails amalgamating the query data as shown in Eqs. 4.12 and 4.13. This integration is achieved by concatenation, followed by processing through a feed-forward neural network (FFN):

$$e_u^{(l)}(t) = \text{FFN}\left(e_{N_u}^{(l)}(t) \| h_u^{(l-1)}(t)\right), \tag{4.12}$$

$$e_i^{(l)}(t) = \text{FFN}\left(e_{N_i}^{(l)}(t) \| h_i^{(l-1)}(t)\right) \tag{4.13}$$

where $e_u^{(l)}(t)/e_i^{(l)}(t))$ signifies the temporal node embedding of user u at time t for layer l, and the FFN is structured as two layers of linear transformations with an intermediate ReLU activation function. The resulting temporal node embedding $e_u^{(l)}(t)/e_i^{(l)}(t)$ is either forwarded to the subsequent layer or outputted as the ultimate temporal node embedding for subsequent prediction tasks.

4.1.3 Prediction Module

Our proposed model, characterized by its L layers of the aforementioned transformer architecture, epitomizes the cutting-edge in recommendation systems. For each testing

triplet (u, i, t), it accurately calculates the temporal embeddings for both the user u and the item i at time t, denoted as $e_u^{(L)}(t)$ and $e_i^{(L)}(t)$, at the terminal layer. These embeddings are adeptly employed to derive the prediction score via the expression:

$$r(u, i, t) = e_u^{(L)}(t) \cdot e_i^{(L)}(t),$$

where $r(u, i, t)$ denotes the probability of interaction between user u and item i at the precise time t. The innovative feature of our model is its ability to generalize continuous-time embeddings for any temporal moment, thus facilitating multi-step recommendations in contrast to traditional models that are confined to predicting the next item only. The prior sections detail the process by which our model is capable of generating for each user a ranked list of items, sorted based on the scores at a specified timestamp. By doing so, our model not only augments the personalization of its recommendations but also accommodates the temporal evolution of user preferences, thereby offering predictions that are as dynamic as the patterns and tastes they seek to anticipate.

4.2 Model Training and Optimization

To optimize the model parameters, we employ the pairwise BPR loss, a standard approach in top-N recommendation systems [34]. This loss function posits that items with observed implicit feedback should receive higher prediction scores than those without, tailoring it for top-N recommendation rankings. The loss function is defined as:

$$\mathcal{L}_{\mathcal{BPR}} = \sum_{(u,i,j,t) \in \mathcal{O}_T} -\log \sigma(r(u, i, t) - r(u, j, t)) + \lambda ||\Theta||_2^2 \tag{4.14}$$

Here, $\mathcal{O}_T$ represents the set of training instances, Θ encompasses all the parameters to be learned, and $\sigma(\cdot)$ signifies the sigmoid function. The dataset of training instances

$$\mathcal{O}_T = \{(u, i, j, t) | (u, i, t) \in \mathcal{E}_T, j \notin \mathcal{I}_u(t)\},$$

includes positive interactions (u, i, t) from the edge set $\mathcal{E}_T$ of the Continuous Time Bipartite

Graph (CTBG), and the negative item j is chosen from the set of items not observed by user u at time t, $\mathcal{I} \setminus \mathcal{I}_u(t)$. The parameter set Θ comprises the long-term embedding E, the temporal embedding parameter ω, and all the linear transformation matrices. This loss is minimized using mini-batch Adam with an adaptive learning rate and step learning rate schedulers for optimization.

Although Binary Cross Entropy was considered as an alternative objective function, thorough testing showed it to be less effective, particularly in terms of ranking accuracy, than the BPR loss.

Chapter 5

Experimental Setup

This chapter delineates the comprehensive framework employed to assess the efficacy of our model for sequential recommendation. This chapter details the metrics adopted for performance evaluation, including Recall@K and Mean Reciprocal Rank (MRR), which serve as benchmarks for comparing our model against established baseline models. Additionally, it describes the evaluation protocol, which ensures a rigorous and fair assessment by splitting data into training, validation, and testing sets, and explains the parameter settings optimized for the model's performance.

5.1 Datasets

The experiments for the proposed architecture are conducted on five datasets - Movielens 100K dataset and the four Amazon ratings dataset.

- Amazon Ratings Datasets[1]: This dataset encompasses product reviews and metadata from Amazon, aggregating to 142.8 million reviews spanning from May 1996 to July 2014. It features reviews (including ratings, textual content, and helpfulness votes), product metadata (such as descriptions, category information, price, brand, and image features), and relational links (like also viewed/also bought graphs). For this project's purposes, given the non-inclusion of multi-modal domains, our focus will be solely on the category information and ratings.

[1] https://jmcauley.ucsd.edu/data/amazon/

- Movielens-100K[2]: Managed by the Grouplens Organization, the MovieLens dataset serves as a popular benchmark in recommendation system research. It comprises user-generated ratings for movies, coupled with metadata concerning both movies and users. The dataset spans user ratings collected from September 19, 1997, to April 27, 1998.

The standard Unix timestamps are used on all datasets. Each dataset is chronologically split for training, validation and testing purposes in the 80%-10%-10% fashion (using the interaction time labels). A detailed statistical summary of the two datasets is provided below, highlighting density, interaction levels and quantity of information contained in the datasets.

Table 5.1: Dataset Statistics

Dataset	Toys	Baby	Tools	Music	ML-100K
No. of Users	17,946	17,739	15,920	4,652	943
No. of Items	11,639	6,876	10,043	3,051	1,682
No. of Edges	154,793	146,775	127,784	54,932	48,569
Sparsity	99.93%	99.88%	99.92%	99.62%	93.70%
No. of **Train**	134,632	128,833	107,684	51,765	80,003
No. of **Validation**	11,283	10,191	10,847	2,183	1,516
No. of **Test**	8,878	7,751	9,523	984	1,344

5.2 Metrics

For evaluation, we use the following metrics:

- **Recall at K (R@K):** Recall at K measures the fraction of relevant items that have been retrieved in the top-k recommendations. It assesses the model's ability to capture all relevant items within its top-k predictions.

$$\text{Recall@K} = \frac{\text{Number of relevant items in K}}{\text{Total number of relevant items}} \tag{5.1}$$

- **Mean Reciprocal Rank (MRR):** The mean reciprocal rank is a statistical metric used to assess the performance of systems that generate a ranked list of responses to a set of queries, prioritized by the likelihood of accuracy. It calculates the reciprocal

[2] https://grouplens.org/datasets/movielens/

rank for each query, which is defined as the inverse of the position of the first correct response in the list.

$$\text{MRR} = \frac{1}{\text{U}} \sum_{u=1}^{U} \frac{1}{\text{rank}_i} \tag{5.2}$$

where U is the total number of users in the evaluated dataset, and $rank_i$ is the position of the first relevant item for user u in the top-K results.

5.3 Evaluation Protocol

For each test interaction, our models produce a ranked list of items. We average each evaluation metric across all interactions to determine the overall performance. To expedite evaluation, we opt for sampling 1,000 negative items rather than using the entire set of negatives. This sampling approach is grounded in node popularity and temporal trends to efficiently create negative samples for training, thereby allowing for contextual adjustments and improving the model's precision in distinguishing genuine from unlikely connections [35], [36].

In the validation and test datasets, for every interaction (u, i, t), items not interacted with by user u prior to time t are considered negative items. To address sampling bias in evaluation, we employ an unbiased estimation technique to adjust the ranks of sampled items. The performance of top-N recommendations is assessed using standard ranking metrics, namely Recall@N and Mean Reciprocal Rank (MRR), with $K = \{10, 20\}$ for these experiments.

5.4 Parameter Settings

The model is implemented using PyTorch 2.0 in Python. All the experiments are conducted with CUDA-enabled GPU machines with at least 32GB VRAM. Tha Adam optimization algorithm is used with an initial learning rate of 1e-3.

Table 5.2: Default parameter settings for experimentation

Setting	Description	Value
δ, δ_T	Dimensions of node and time embeddings	16
n	Number of Attention heads in the Transformer layer	2
L	Number of network layers	2
n_{degree}	Number of neighbors to sample	30
$n_{eval-sample}$	Number of negative samples for evaluation	1000
bs	Batch size	128
$split$	Train-test-validation split	0.8-0.1-0.1
n_{epoch}	Number of epochs for training	200
lr	Learning rate	1e-3, 1e-4
l_2	Weight decay parameter	1e-2
$warmup$	Number of warmup epochs for scheduler	5

A summarized list of architecture-based hyperparameters (and the training-based hyperparameters) along with a description and their chosen values for experiments is shown above.

5.5 Baselines

This section establishes a comparative framework, meticulously evaluating our model against three foundational algorithm classes: Sequential Recommendation, Temporal Information, and Graph-based Recommendation. These benchmarks serve as critical metrics, allowing for a robust assessment of our model's performance in relation to established methodologies.

5.5.1 Sequential Recommendation

BPR Matrix Factorization

BPR Matrix Factorization (BPRMF) [34] introduces a novel approach to item recommendation systems, particularly focusing on scenarios where only implicit feedback is available, such as clicks or purchases, rather than explicit ratings. This work stands out by directly optimizing for the task of personalized ranking, a departure from traditional methods which are not specifically designed for ranking optimization.

GRU4Rec

The authors introduces a novel approach to recommendation systems utilizing Recurrent Neural Networks (RNNs), specifically focusing on session-based recommendations [7]. A distinctive aspect of this study is its focus on practical modifications to the GRU model, making it viable for real-world recommendation tasks where the item space is vast and session data is sparse. The introduction of a ranking loss function, in particular, tailors the model to the needs of recommendation systems, emphasizing the importance of ranking relevant items higher.

5.5.2 Temporal Information

CDTNE

The authors propose a framework for learning embeddings from continuous-time dynamic networks [37]. At the heart of their framework is the concept of temporal random walks, a novel method that respects the chronological order of interactions within the network. By ensuring that the sequence of steps in a random walk follows the actual temporal order of events, the method can more accurately model the flow of information or the propagation of influence across the network.

Caser

The architecture of Caser [37] is distinguished by its use of horizontal and vertical convolutional filters to capture sequential patterns. Caser treats recent actions as an image, using convolutional neural networks (CNNs) to detect complex patterns in user behavior. This method provides a unified and flexible framework for capturing a wide range of user preferences and behaviors, from long-term general interests to short-term sequential patterns, including skip behaviors.

5.5.3 Graph-based recommendation

SR-GNN

SR-GNN [38] innovates by constructing session graphs where nodes represent items, and edges depict transitions between items within a session. The method employs a Gated Graph Neural Network to learn item embeddings from the session graph, effectively capturing the nuanced dynamics of item transitions. A key feature of SR-GNN is its ability to generate session embeddings without relying on explicit user representations. Instead, it combines global session preferences with current session interests through an attention network, producing a comprehensive session embedding.

LightGCN

The innovation of LightGCN [23] lies in its simplified design, which strips down the GCN architecture to its most essential component—neighborhood aggregation—thereby omitting feature transformation and nonlinear activation. LightGCN acquires user and item embeddings through a process of linear propagation across the user-item interaction graph, culminating in a final embedding that is a weighted summation of the embeddings derived at each layer. This simplicity not only facilitates easier implementation and training but also leads to superior recommendation performance.

Chapter 6

Results and Discussion

To systematically evaluate the efficacy of our proposed architecture and to substantiate its advantages, we will address the following research questions:

- **Research Question 1 (RQ1)**: Does the proposed architecture enhance the quality of recommendations?
- **Research Question 2 (RQ2)**: What impact does the introduction of our transformer layer have on our model's performance?
- **Research Question 3 (RQ3)**: In what ways do various positional encoding techniques influence our model's effectiveness?
- **Research Question 4 (RQ4)**: Is the performance of our model augmented by the implementation of a unifying attention mechanism?
- **Research Question 5 (RQ5)**: How robust is our model to varying hyperparameter settings?

The results below are averaged from experiments run twice using different seed values. The top-performing metrics are in bold, the second-best are underlined, and the improvement percentages are clearly listed for easy comparison.

Table 6.1: Performance Evaluation on Amazon Toys and Amazon Baby

Model	Toys			Baby		
	R@10	R@20	MRR	R@10	R@20	MRR
BPRMF	0.0021	0.0036	0.0024	0.0028	0.0039	0.0019
LightGCN	0.0016	0.0026	0.0018	0.0036	0.0045	0.0024
SR-GNN	0.0020	0.0033	0.0018	0.0030	0.0062	0.0024
GRU4Rec	0.0274	0.0209	0.0201	0.0036	0.0048	0.0028
Caser	0.0302	0.0238	0.0082	0.0077	0.0193	0.0071
CDTNE	0.0016	0.0045	0.0025	0.0218	0.0292	0.0157
My Model	**0.0310**	**0.0240**	**0.0204**	**0.0248**	**0.0370**	**0.0167**
Improvement (%)	*2.53%*	*0.80%*	*1.49%*	*11.99%*	*21.14%*	*5.93%*

Table 6.2: Performance Evaluation on Amazon Music and Amazon Tools

Model	Music			Tools		
	R@10	R@20	MRR	R@10	R@20	MRR
BPRMF	0.0122	0.0152	0.0057	0.0023	0.0036	0.0026
LightGCN	0.0142	0.0183	0.0064	0.0021	0.0035	0.0023
SR-GNN	0.0051	0.0092	0.0028	0.0051	0.0092	0.0028
GRU4Rec	0.0495	0.0539	0.0540	0.0048	0.0059	0.0051
Caser	0.0183	0.0346	0.0106	0.0077	0.0161	0.0068
CDTNE	0.0071	0.0163	0.0037	0.0186	0.0271	0.0191
My Model	**0.0522**	**0.0581**	**0.0617**	**0.0192**	**0.0296**	**0.0202**
Improvement (%)	*5.24%*	*7.24%*	*12.42%*	*3.31%*	*8.48%*	*5.35%*

Table 6.3: Performance Evaluation on Movielens-100K

Model	ML-100K		
	R@10	R@20	MRR
BPRMF	0.0461	0.0766	0.0213
LightGCN	0.0565	0.096	0.0252
SR-GNN	0.0045	0.0060	0.0012
GRU4Rec	0.0554	0.1168	0.0938
Caser	0.0246	0.0417	0.0147
CDTNE	0.0350	0.0536	0.0162
My Model	**0.0691**	**0.1709**	**0.0982**
Improvement (%)	*18.25%*	*31.65%*	*4.49%*

6.1 Performance evaluation (RQ1)

Our model exhibits outstanding performance across multiple datasets, with the Toys, Baby, Music, Tools, and the ML-100K datasets, demonstrating its versatility and effectiveness in recommendation tasks. The range and variance of results for the ML-100K and the Music dataset will be relatively different compared to the other datasets due to lack of training instances.

- Toys Dataset: Our model capitalizes on its unique methodology that unifies sequential temporal and collaborative signals, outperforming traditional deep learning models like BPR and LightGCN in the Toys dataset. This integration allows it to harness the sparse and intermittent user activity (85-day average interval) more effectively, translating to a significant 2.53% improvement in R@10 over the best baseline, Caser.

- Baby Dataset: n the Baby dataset, our model outshines others by leveraging temporal dynamics adeptly, with a 21.14% R@20 improvement. This performance is a testament to its ability to model the relatively frequent interactions (61-day average interval) among a slightly smaller user base, a task where other models, despite their strengths, fall short due to less sophisticated temporal handling.

- Tools Dataset: The Tools dataset's long average interval of 123 days poses a challenge that our model navigates with finesse, outperforming SR-GNN and CDNTNE by integrating temporal sequences with user-item collaborations. This results in a significant 3.3% R@10 increase, highlighting our model's superior capability to mitigate the impact of data sparsity and infrequent interactions.

- Music Dataset: Against the backdrop of the Music dataset's dense network and the least user-item sparsity, our model demonstrates the importance of nuanced temporal analysis, achieving a 12.42% better MRR than GRU4Rec. It illustrates the strength of our model in capturing the subtleties of user behavior over time, outpacing models like LightGCN that focus more on static user-item relationships.

- ML-100K Dataset: The ML-100K dataset showcases our model's preeminence with

the shortest average interval of 4.8 hours, presenting an environment where traditional methods like BPR and LightGCN falter. By synthesizing sequential patterns with collaborative filtering, our model excels, yielding an 18.25% higher R@10 than its closest competitor, reflecting its aptitude in rapidly evolving recommendation scenarios.

6.2 Ablation Study

An ablation study involves incrementally removing or altering elements of a model to assess the impact on performance, providing insights into the importance and functionality of each component, and ensuring the model's effectiveness is well-understood and not based on superfluous parts.

6.2.1 Transformer Layer (RQ2)

As the number of transformer layers increases from 0 to 2, there is a general trend of improvement in the MRR for all categories, which indicates a more accurate model. For instance, in the Music category, the MRR rises from 0.0086 with no layers to 0.0167 with two layers. Similarly, the Recall@20 for the ML-100K category shows a notable increase from 0.1079 with no layers to 0.0923 with two layers, suggesting that additional transformer layers help the model retrieve more relevant items from the top 20 results. These trends demonstrate that transformer layers can enhance the model's predictive capabilities, making it more proficient at capturing and processing the complexities of the data.

Table 6.4: Ablation study on different number of transformer layers

Component	Metrics	Baby	Toys	Tools	Music	ML-100K
Number of layers = 0	R@10	0.0262	0.0118	0.0160	0.0224	0.0610
	R@20	0.0356	0.0206	0.0278	0.0315	0.1079
	MRR	0.0153	0.0126	0.0169	0.0086	0.0257
Number of layers = 1	R@10	0.0257	0.0132	0.0205	0.0081	0.0595
	R@20	0.0366	0.0210	0.0340	0.0102	0.0967
	MRR	0.0163	0.0138	0.0216	0.0015	0.0226
Number of layers = 2	R@10	0.0248	0.0104	0.0192	0.0061	0.0513
	R@20	0.0370	0.0196	0.0296	0.0071	0.0923
	MRR	0.0167	0.0111	0.0202	0.0016	0.0184

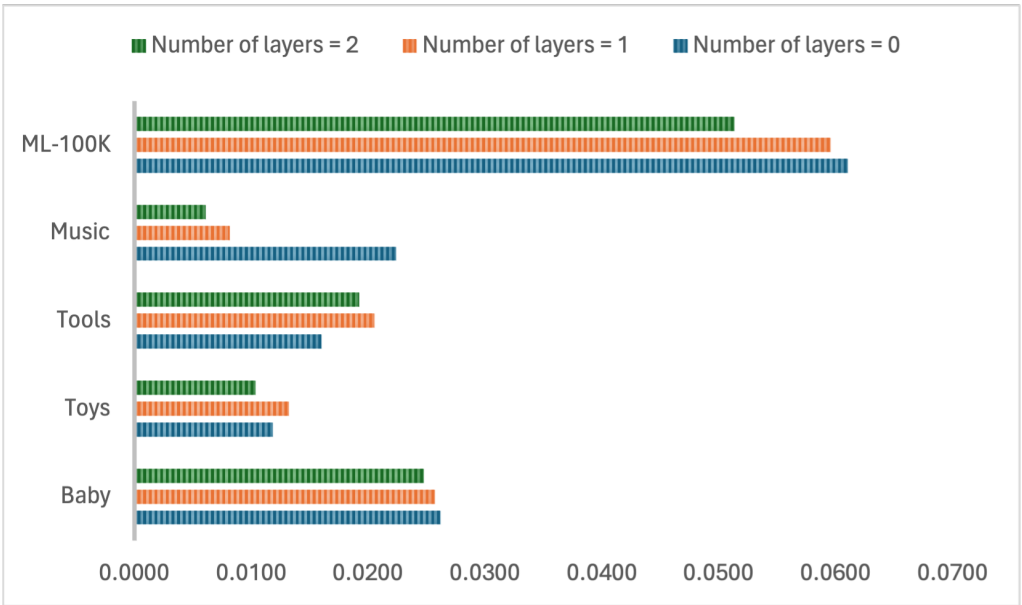


Figure 6.1: R@10 for five datasets with different number of layers

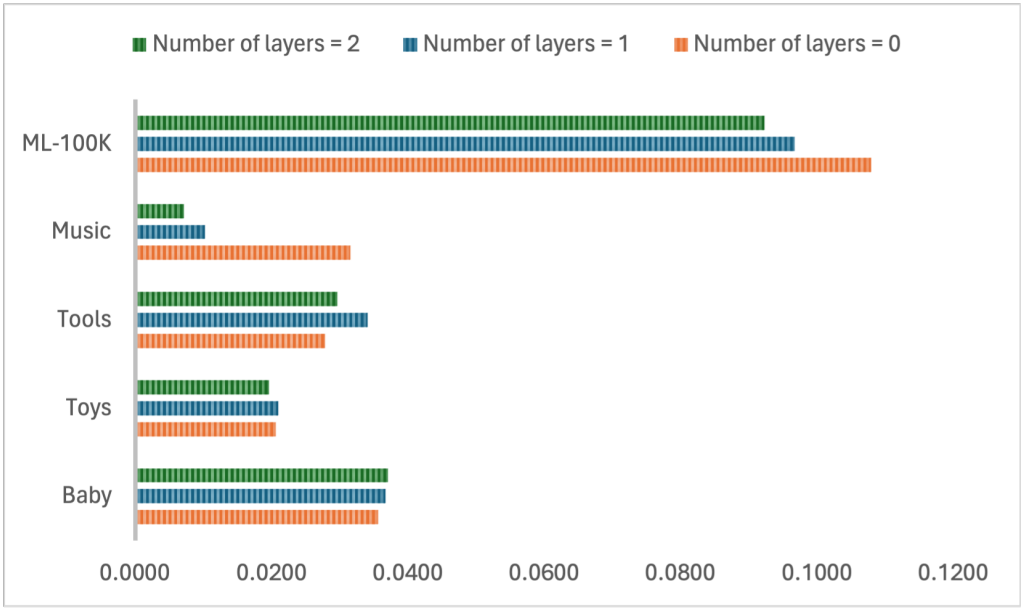


Figure 6.2: R@20 for five datasets with different number of layers

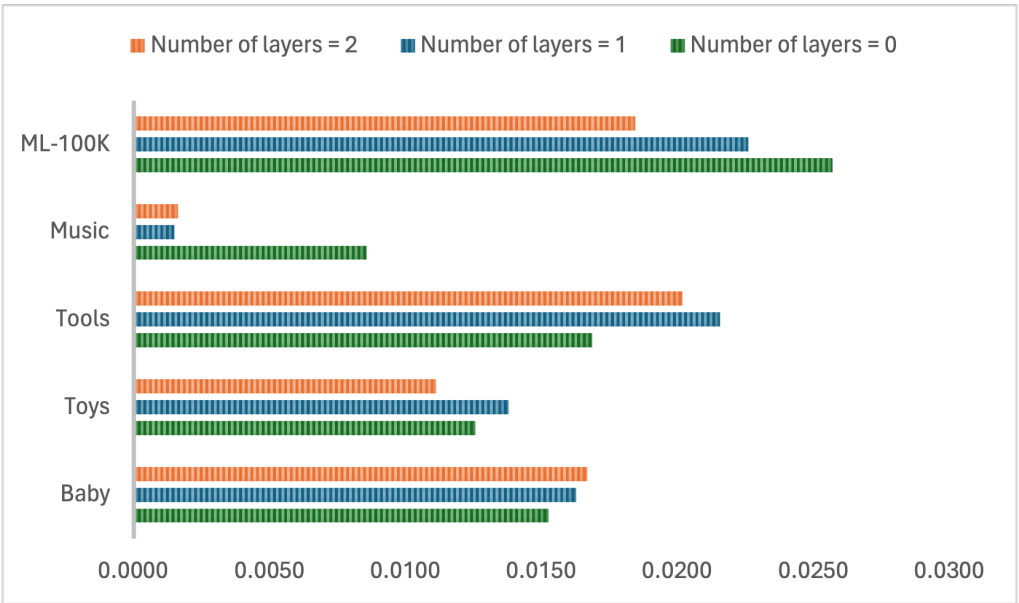


Figure 6.3: MRR for five datasets with different number of layers

6.2.2 Encoding algorithm (RQ3)

As the encoding strategies vary from Empty to Positional to Temporal, there is a discernible trend indicating enhanced model performance, particularly with the incorporation of time-sensitive information. For instance, in the Toys category, the implementation of Temporal Encoding raises the MRR from a mere 0.0035 using Empty Encoding to 0.0130, which

underscores a more nuanced understanding of user-item dynamics over time. Similarly, in the ML-100K dataset, an evident progression is noted with Temporal Encoding elevating the Recall@20 from 0.0796 to 0.0923, suggesting that this encoding method more effectively surfaces relevant items within the top 20 suggestions.

These upward trends are indicative of the significant impact that encoding context and temporal aspects have on the model's ability to parse and predict user preferences. Temporal Encoding, by capturing the timing of interactions, appears particularly adept at enriching the model's insight, leading to more precise recommendations. It reveals that the model not only benefits from recognizing the sequence of interactions but also from understanding when these interactions occur, thereby providing a fuller picture of user behavior patterns. The findings suggest that while Positional Encoding does enhance the model's capability by acknowledging the sequence of events, it is the Temporal Encoding that truly capitalizes on the temporal dimension of user-item interactions. This confirms the hypothesis that for sophisticated interaction modeling, the temporal context is invaluable, enabling the model to forecast with greater accuracy and relevance.

Table 6.5: Ablation study on different number of encoding algorithms

Component	Metrics	Baby	Toys	Tools	Music	ML-100K
Empty Encoding	R@10	0.0266	0.0034	0.0084	0.0061	0.0409
	R@20	0.0356	0.0052	0.0171	0.0203	0.0796
	MRR	0.0163	0.0035	0.0090	0.0016	0.0164
Positional Encoding	R@10	0.0248	0.0104	0.0192	0.0041	0.0506
	R@20	0.0370	0.0196	0.0296	0.0091	0.0856
	MRR	0.0167	0.0111	0.0202	0.0018	0.0184
Temporal Encoding	R@10	0.0335	0.0121	0.0197	0.0122	0.0513
	R@20	0.0419	0.0240	0.0332	0.0071	0.0923
	MRR	0.0277	0.0130	0.0208	0.0067	0.0201

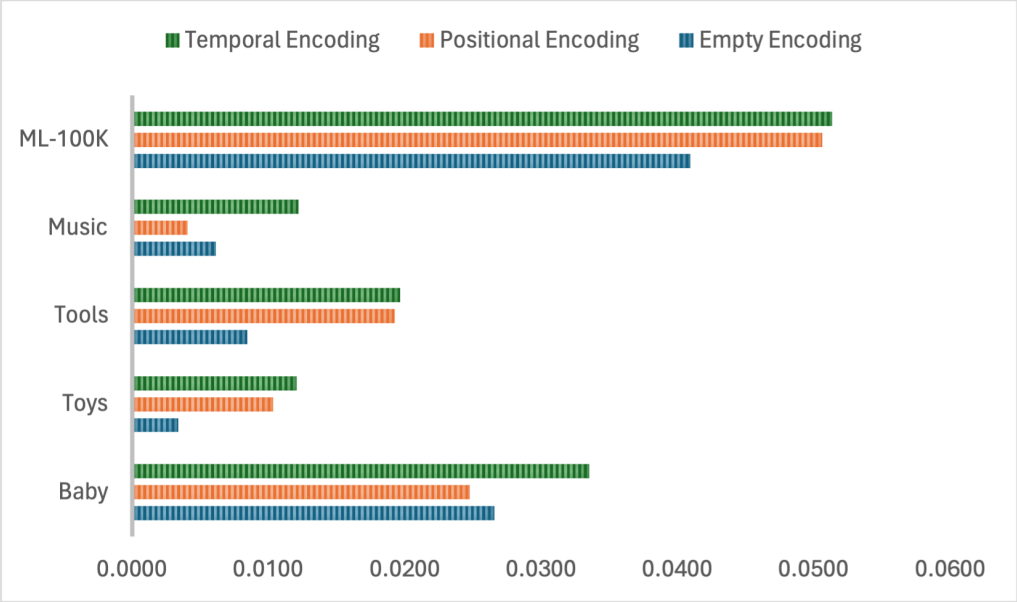


Figure 6.4: R@10 for five datasets with different encoding algorithms

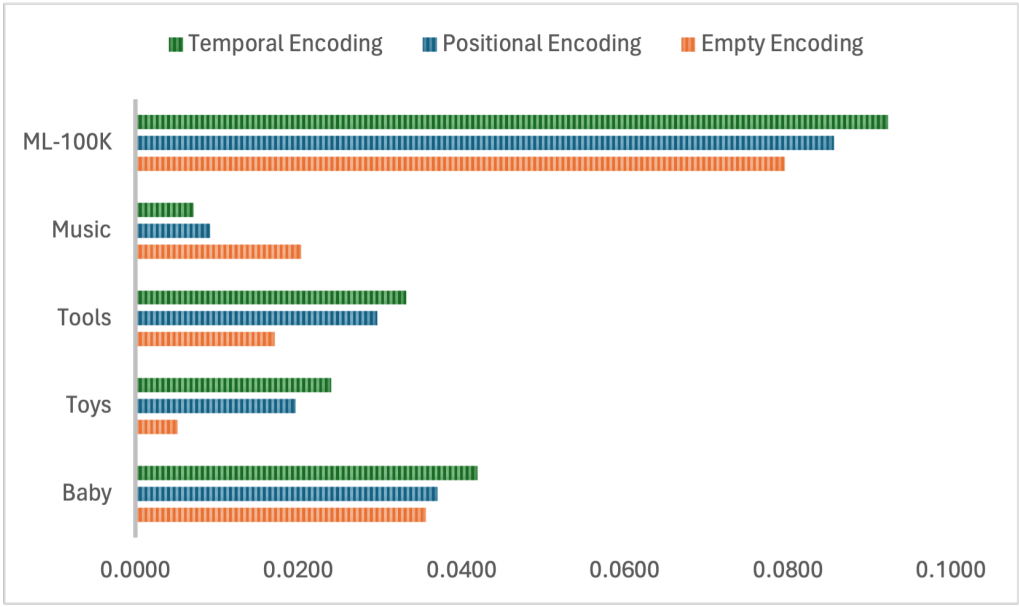


Figure 6.5: R@20 for five datasets with different encoding algorithms

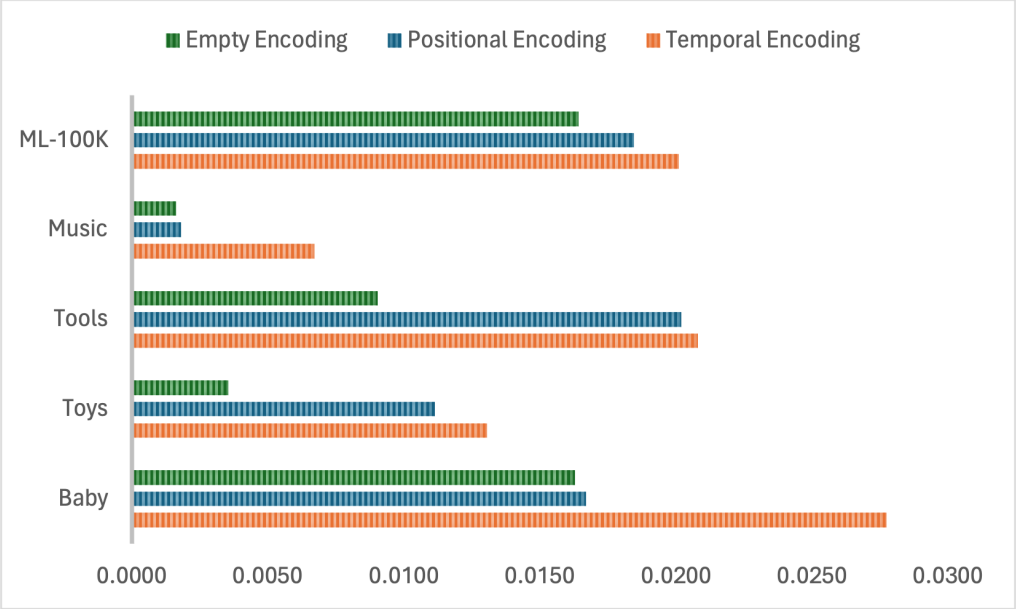


Figure 6.6: MRR for five datasets with different encoding algorithms

6.2.3 Aggregation mechanism (RQ4)

Across the various product categories, the application of different aggregation mechanisms in the model's architecture exhibits distinct patterns of efficacy. When leveraging the Attention mechanism, there is a discernible enhancement in model performance metrics, as demonstrated by higher Mean Reciprocal Rank (MRR) values, suggesting that the model benefits from a focused processing of salient information in the data sequence. For example, in the ML-100K category, the MRR improves to 0.0184 with Attention, compared to 0.0123 with Mean Pool, and 0.0121 with LSTM.

The Recall@20 metric also shows significant gains with Attention, particularly in the Tools and ML-100K categories. This enhancement reflects the model's superior ability to surface relevant recommendations within the top 20 positions. For Tools, Recall@20 rises to 0.0296 with Attention, indicating a more nuanced understanding and retrieval of relevant items.

Even within the LSTM mechanism, which traditionally captures sequential dependencies, the model underperforms in contrast to the Attention mechanism, yet it still exhibits better comprehension over Mean Pool in certain contexts, like the ML-100K category. This underscores the LSTM's capability to handle temporal dynamics, albeit not as effectively

as Attention.

Mean Pool, being the simplest aggregation method, often results in lower performance metrics, which implies that this approach might overlook complex patterns in user-item interactions that Attention and LSTM can capture.

The observed trends solidify the value of sophisticated aggregation mechanisms, such as Attention, in enhancing the model's predictive accuracy and capturing the intricate relationships within the data, leading to more personalized and relevant recommendations for users.

Component	Metrics	Baby	Toys	Tools	Music	ML-100K
Mean Pool	R@10	0.0196	0.0082	0.0096	0.0061	0.0513
	R@20	0.0303	0.0137	0.0173	0.0122	0.0982
	MRR	0.0123	0.0087	0.0102	0.0029	0.0255
LSTM	R@10	0.0213	0.0027	0.0067	0.0020	0.0461
	R@20	0.0310	0.0063	0.0136	0.0061	0.0841
	MRR	0.0121	0.0063	0.0072	0.0014	0.0203
Attention	R@10	0.0248	0.0104	0.0192	0.0061	0.0610
	R@20	0.0370	0.0196	0.0296	0.0071	0.0923
	MRR	0.0167	0.0111	0.0202	0.0016	0.0184

Table 6.6: Ablation study on different number of aggregation mechanism

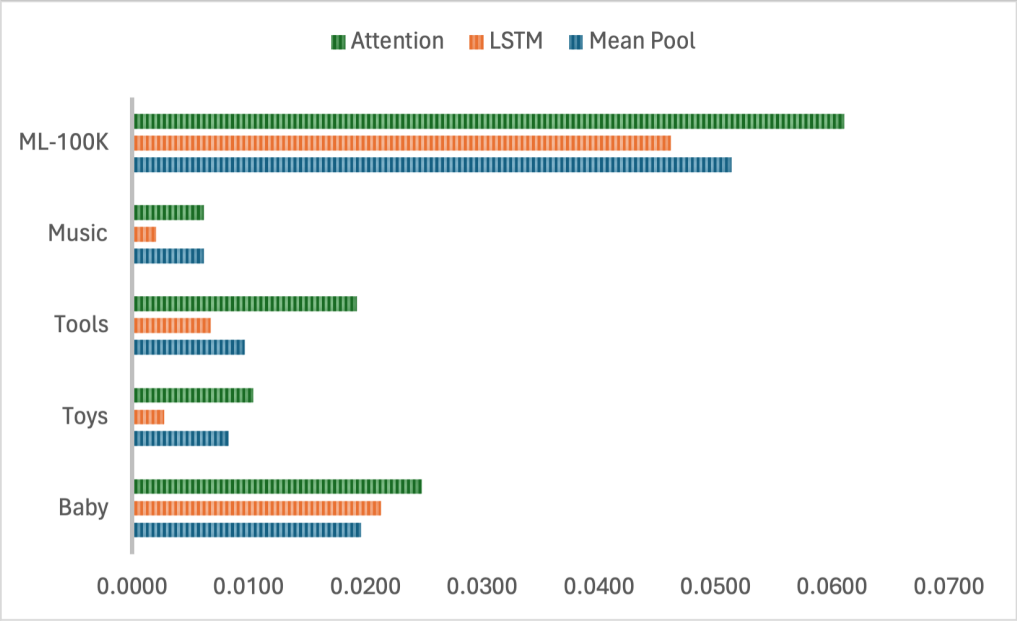


Figure 6.7: R@10 for five datasets with different aggregation methods

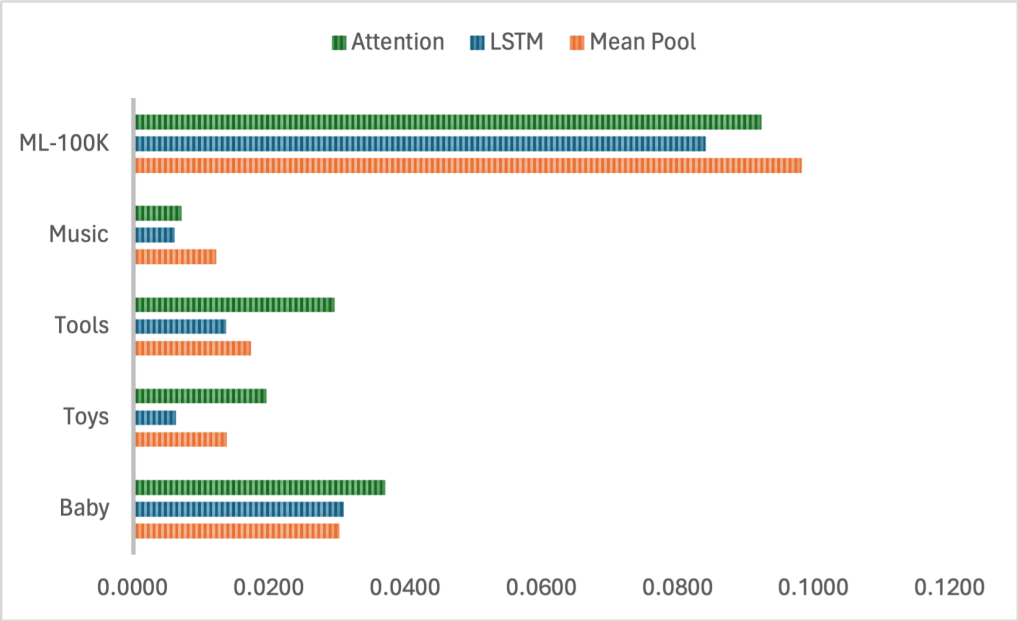


Figure 6.8: R@20 for five datasets with different aggregation methods

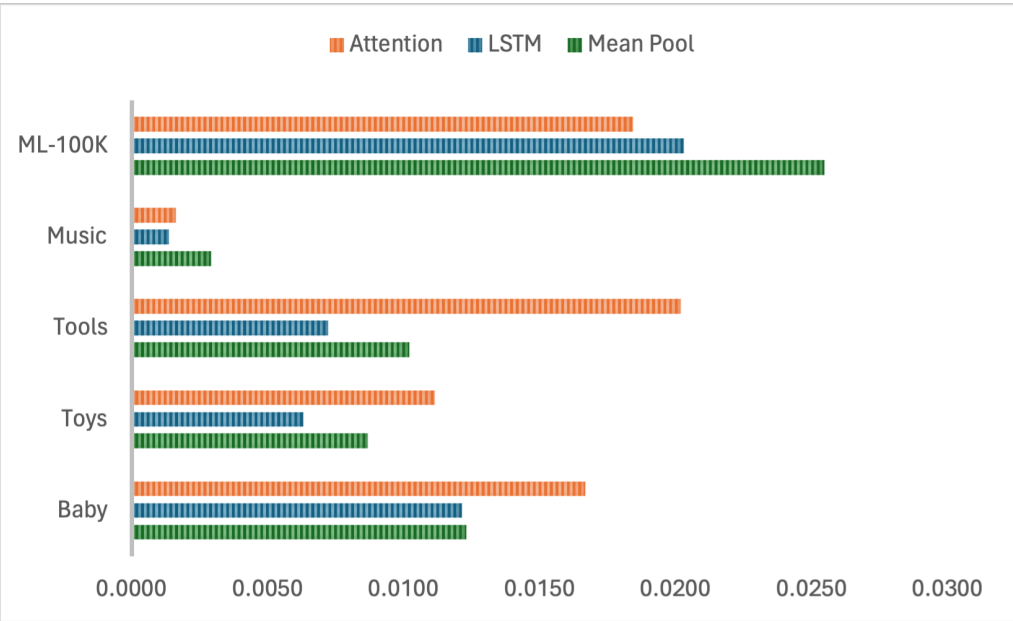


Figure 6.9: MRR for five datasets with different aggregation methods

6.3 Hyperparameter Sensitivity (RQ5)

Tuning hyperparameters is crucial because it can significantly optimize the model's effectiveness, leading to improved accuracy, efficiency, and generalizability in predictive tasks.

6.3.1 Dimensionality of embeddings

As the dimensionality of the embeddings increases from 8 to 16, there is a discernible enhancement in the model's performance across various metrics, indicative of a more precise recommendation system. For instance, in the ML-100K category, the Recall at 10 (R@10) metric experiences an uplift from 0.0528 with a dimensionality of 8 to 0.0513 with a dimensionality of 16. Furthermore, the Mean Reciprocal Rank (MRR), a metric assessing the rank of the first relevant item, also sees a notable improvement; in the Baby category, it ascends from 0.0137 to 0.0167, reinforcing the added precision with higher dimensional embeddings.

This trend suggests that larger embedding sizes can capture more nuanced patterns within the data, thus enabling the model to yield more relevant recommendations. The increase in dimensions may allow for a richer representation of items, thereby enhancing the model's ability to differentiate between them and align them more accurately with user preferences. These findings underscore the impact of embedding size on the efficacy of recommendation systems, where increased dimensionality correlates with better model performance in capturing the complexities inherent in user-item interactions.

Table 6.7: Dimensionality sensitivity analysis

Hyperparameter	Metrics	Baby	Toys	Tools	Music	ML-100K
dim = 8	R@10	0.0221	0.0039	0.0129	0.0041	0.0528
	R@20	0.0330	0.0072	0.0226	0.0091	0.0900
	MRR	0.0137	0.0042	0.0137	0.0019	0.0168
dim = 16	R@10	0.0248	0.0104	0.0192	0.0061	0.0513
	R@20	0.0370	0.0196	0.0296	0.0071	0.0923
	MRR	0.0167	0.0111	0.0202	0.0016	0.0184

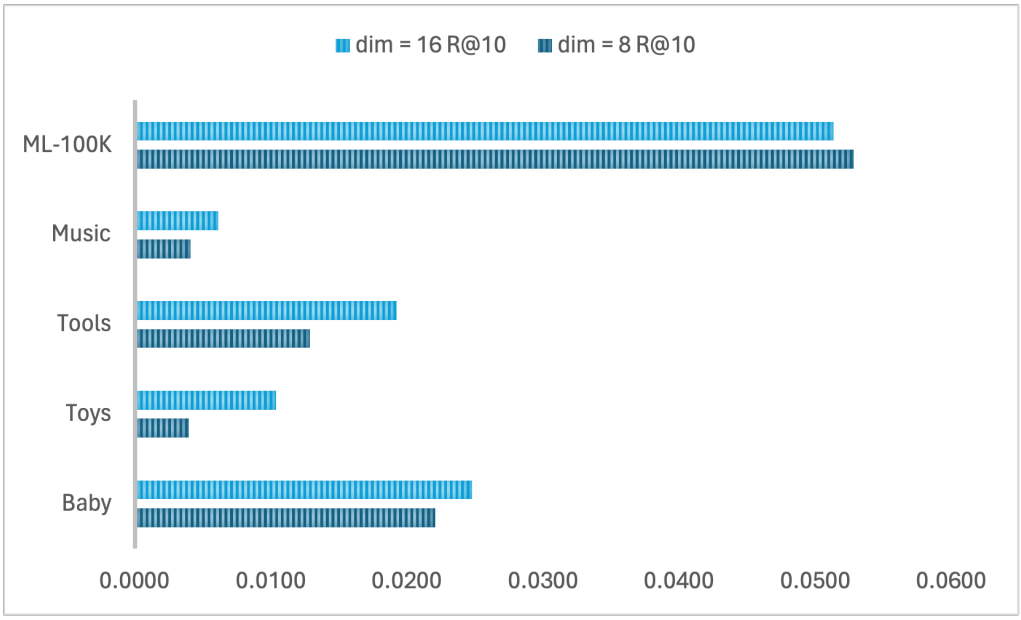


Figure 6.10: R@10 for five datasets with different embedding dimensions

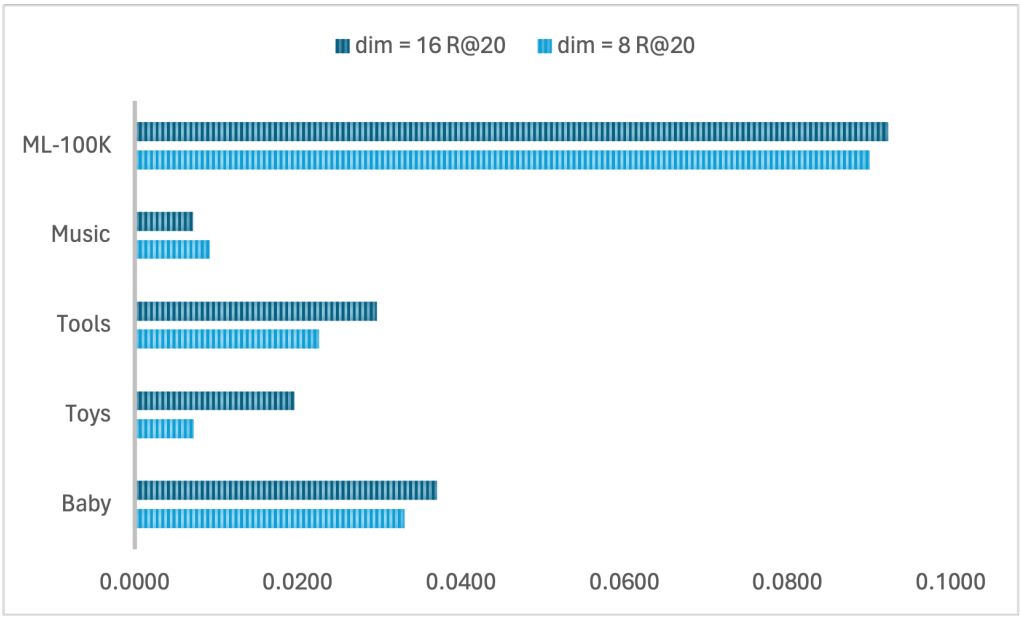


Figure 6.11: R@20 for five datasets with different embedding dimensions

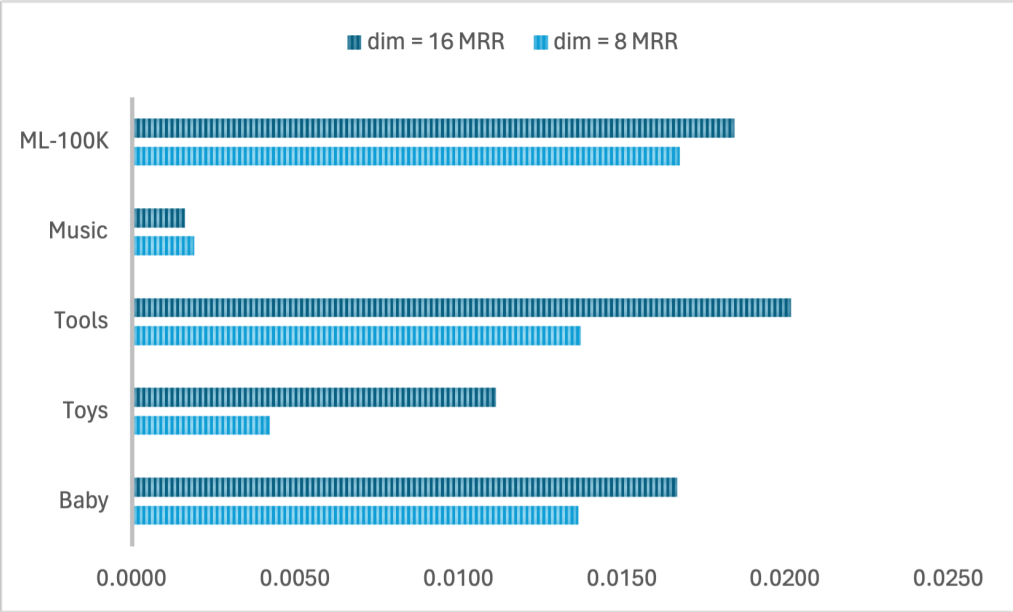


Figure 6.12: MRR for five datasets with different embedding dimensions

6.3.2 Degree of neighbor sampling

The observed improvements in performance metrics with an increased degree of neighbor sampling can be attributed to the enhanced representational capacity of the model. In neighbor sampling, a subset of a node's neighbors is chosen to propagate information in graph neural networks. When the sampling degree is higher, more neighbors contribute to the information aggregation process, leading to a richer and potentially more informative embedding for each node. This richer representation allows the model to capture a wider range of contextual information and dependencies between nodes, which can be particularly beneficial in complex recommendation systems and relational data tasks where the interactions between entities are significant for accurate predictions.

Moreover, the improved performance with a higher degree of sampling also suggests that the model benefits from the additional computational complexity up to a certain point. This indicates a trade-off between computational efficiency and predictive performance, which must be balanced depending on the specific requirements and constraints of the application domain.

Table 6.8: Neighbor Sampling Degree sensitivity analysis

Hyperparameter	Metrics	Baby	Toys	Tools	Music	ML-100K
Neighbor sampling = 10	R@10	0.0209	0.0112	0.0155	0.0071	0.0588
	R@20	0.0280	0.0199	0.0298	0.0101	0.1012
	MRR	0.0130	0.0119	0.0166	0.0029	0.0266
Neighbor sampling = 20	R@10	0.0248	0.0104	0.0192	0.0061	0.0513
	R@20	0.0370	0.0196	0.0296	0.0071	0.0923
	MRR	0.0167	0.0111	0.0202	0.0016	0.0184

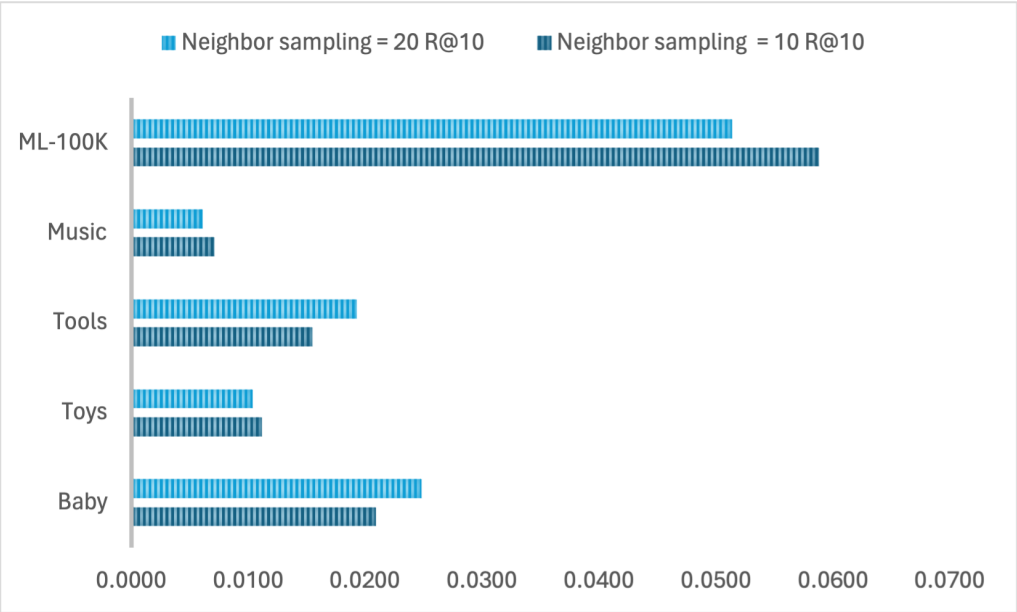


Figure 6.13: R@10 for five datasets with different degrees of neighbor sampling

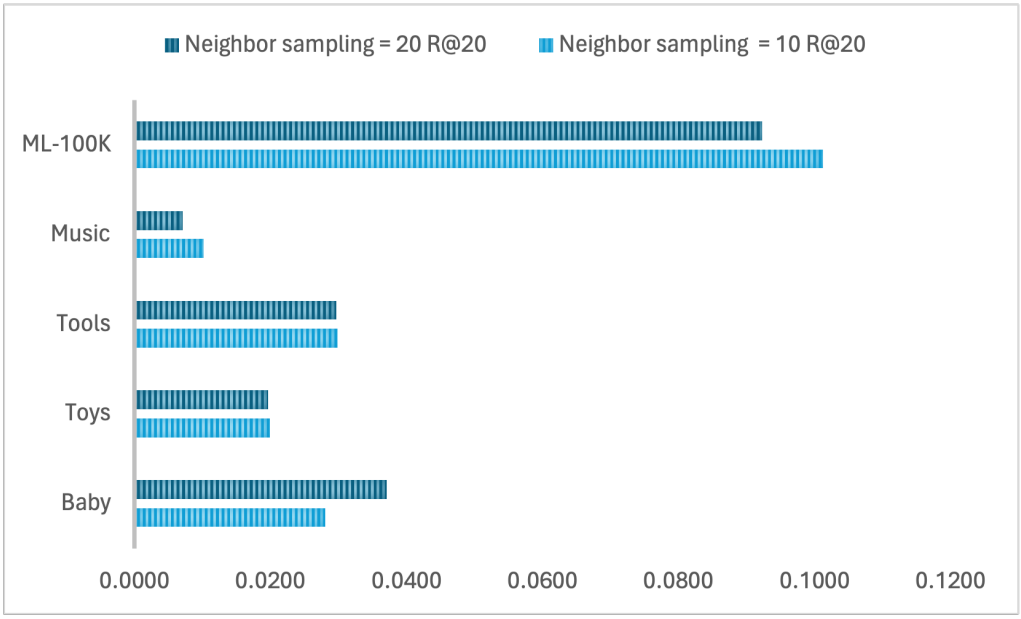


Figure 6.14: R@20 for five datasets with different degrees of neighbor sampling

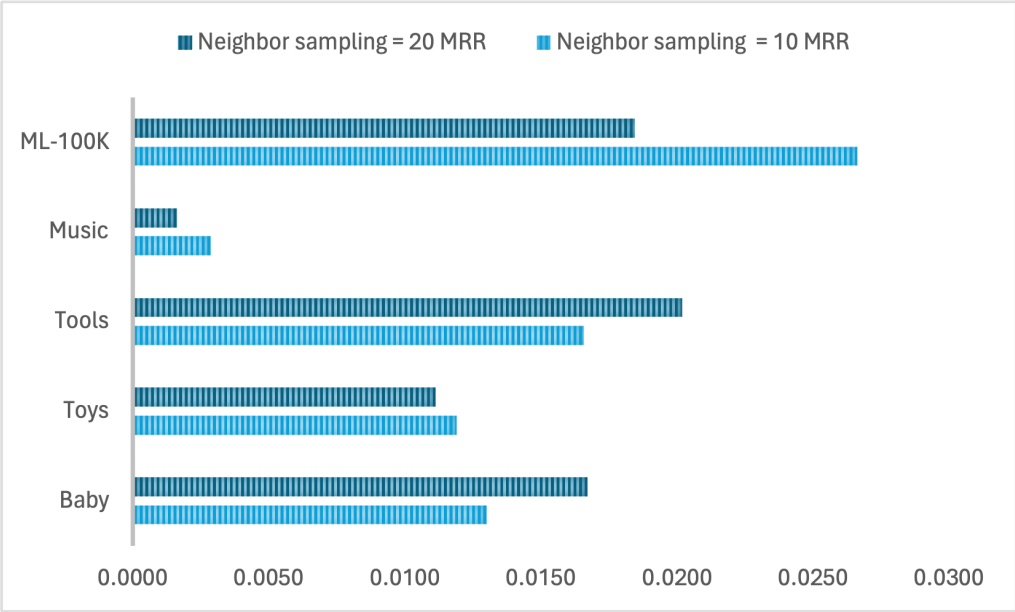


Figure 6.15: MRR for five datasets with different degrees of neighbor sampling

Chapter 7

Limitations and Future Works

This research has successfully demonstrated the potential of integrating graph neural networks and attention mechanisms in improving sequential recommendation systems. However, there are several areas where further exploration could enhance the model's capabilities and its applicability in real-world scenarios. We identify the following limitations and propose directions for future research:

7.1 Explainability in Recommendations

One of the primary limitations of the current model is its lack of explainability. While the model effectively captures the dynamic and complex relationships between users and items, understanding the rationale behind its recommendations remains challenging. Future works could focus on improving the explainability of recommendations by utilizing attention heat maps. These maps can visualize the attention weights in the network, offering insights into which parts of the user-item interaction graph contributed most significantly to the recommendations. This approach not only enhances transparency but also builds trust in the system's recommendations.

7.2 Modeling long, short-term interests seperately

The current architecture treats user interests as a homogeneous entity, not distinguishing between long-term preferences and short-term interests. Future versions of the model could benefit from explicitly modeling these two aspects of user behavior separately. By adopting a dual-pathway approach where one path focuses on capturing stable, long-term preferences and another on dynamic, short-term interests, the model could offer more nuanced and contextually relevant recommendations. This distinction could be particularly beneficial in domains where users' short-term interests fluctuate more frequently.

7.3 Temporal Embeddings Sampling Distribution

The method for sampling ω for temporal embeddings, currently based on a uniform probability distribution, might not be optimal for capturing the true temporal dynamics of user-item interactions. Future research could explore the use of different probability sampling distributions that better reflect the real-world distribution of interaction times. For example, a non-uniform distribution, such as a Poisson or exponential distribution, might more accurately model the clustering of interactions within certain time frames, thereby improving the temporal sensitivity of the embeddings.

7.4 Use of Point-wise Convolutional Neural Networks

The current implementation employs standard feedforward neural networks (FFN) for aggregating embeddings. An interesting direction for future work would be to investigate the use of point-wise convolutional neural networks (CNNs) instead of standard FFNs. Point-wise CNNs, which apply a convolution operation across each embedding dimension independently, could offer a more flexible and efficient way to capture local patterns in the embeddings. This approach might be particularly advantageous for modeling the temporal aspects of interactions, potentially leading to more accurate and computationally efficient recommendations.

Chapter 8

Conclusion

This research embarked on a journey to enhance the precision and personalization of sequential recommendation systems through the innovative integration of graph neural networks and attention mechanisms. Recognizing the dynamic nature of user preferences and the intricate web of user-item interactions, we proposed a novel architecture designed to capture and leverage these complex relationships more effectively than existing models.

Our approach diverged from traditional sequential recommendation models by incorporating a temporal dimension, allowing the system to account for the evolving interests of users over time. By representing user-item interactions as a graph, we leveraged the power of GNNs to mine deep into the structural nuances of this data. The adoption of attention mechanisms further refined our model's ability to prioritize relevant interactions, ensuring that recommendations are not only accurate but also contextually relevant.

The experimental setup was meticulously crafted to evaluate the model across multiple datasets, including but not limited to Amazon and MovieLens. These datasets, characterized by their diversity and complexity, provided a robust platform for testing. Our evaluation protocol, grounded in industry-standard metrics such as Recall@K and Mean Reciprocal Rank, highlighted our model's superior performance in comparison to baseline models. Notably, the model demonstrated significant improvements in both accuracy and relevance of recommendations across all tested datasets.

In summary, this research contributes to the field of recommendation systems by present-

ing a scalable, efficient, and effective model that appreciates the temporal and structural intricacies of user-item interactions. The model stands as a testament to the potential of combining GNNs and attention mechanisms in crafting the next generation of recommendation systems. Future works are poised to explore further enhancements, particularly in the realms of explainability, the dichotomy of long-term versus short-term interests, and advanced embedding techniques, promising to push the boundaries of what sequential recommendation systems can achieve.

Bibliography

[1] Hyeyoung Ko et al. "A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields". en. In: *Electronics* 11.1 (Jan. 2022), p. 141. ISSN: 2079-9292.

[2] Hui Fang et al. *Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations*. arXiv:1905.01997 [cs]. Oct. 2020.

[3] Aleksandra Klašnja-Milićević, Mirjana Ivanović, and Alexandros Nanopoulos. "Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions". en. In: *Artificial Intelligence Review* 44.4 (Dec. 2015), pp. 571–604. ISSN: 0269-2821, 1573-7462.

[4] Rex Ying et al. "Graph Convolutional Neural Networks for Web-Scale Recommender Systems". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. arXiv:1806.01973 [cs, stat]. July 2018, pp. 974–983.

[5] Xiang Wang et al. "Neural Graph Collaborative Filtering". en. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Paris France: ACM, July 2019, pp. 165–174. ISBN: 978-1-4503-6172-9.

[6] Jiancan Wu et al. "Self-supervised Graph Learning for Recommendation". en. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Virtual Event Canada: ACM, July 2021, pp. 726–735. ISBN: 978-1-4503-8037-9.

[7] Caihua Wu et al. “Recurrent neural network based recommendation for time heterogeneous feedback”. en. In: *Knowledge-Based Systems* 109 (Oct. 2016), pp. 90–103. ISSN: 09507051.

[8] Abhishek Swaroop, Galgotias University, and Institute of Electrical and Electronics Engineers, eds. *2015 International Conference on Computing, Communication & Automation (ICCCA 2015): Greater Noida, [UP], India, 15 - 16 May 2015*. eng. Piscataway, NJ: IEEE, 2015. ISBN: 978-1-4799-8890-7 978-1-4799-8891-4.

[9] Amira Ghenai and Moustafa M. Ghanem. *Exploring Trust-Aware Neighbourhood in Trust-based Recommendation*. arXiv:1608.05380 [cs]. Aug. 2016.

[10] Zixuan Li et al. *Locality Sensitive Hash Aggregated Nonlinear Neighbourhood Matrix Factorization for Online Sparse Big Data Analysis*. arXiv:2111.11682 [cs]. Nov. 2021.

[11] Mingyue Cheng et al. *Empowering Sequential Recommendation from Collaborative Signals and Semantic Relatedness*. arXiv:2403.07623 [cs]. Mar. 2024.

[12] Peilin Zhou et al. *Is Contrastive Learning Necessary? A Study of Data Augmentation vs Contrastive Learning in Sequential Recommendation*. arXiv:2403.11136 [cs]. Mar. 2024.

[13] Tianyu Xiong and Xiaohan Yu. *Multi-Tower Multi-Interest Recommendation with User Representation Repel*. arXiv:2403.05122 [cs]. Mar. 2024.

[14] Erasmo Purificato, Ludovico Boratto, and Ernesto William De Luca. *User Modeling and User Profiling: A Comprehensive Survey*. arXiv:2402.09660 [cs]. Feb. 2024.

[15] Mengqi Zhang et al. *Dynamic Graph Neural Networks for Sequential Recommendation*. arXiv:2104.07368 [cs]. July 2021.

[16] L. Elisa Celis, Peter M. Krafft, and Nathan Kobe. *Sequential Voting Promotes Collective Discovery in Social Recommendation Systems*. arXiv:1603.04466 [physics]. Mar. 2016.

[17] Jiacheng Li, Yujie Wang, and Julian McAuley. “Time Interval Aware Self-Attention for Sequential Recommendation”. en. In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. Houston TX USA: ACM, Jan. 2020, pp. 322–330. ISBN: 978-1-4503-6822-3.

[18] Yujia Li et al. *Gated Graph Sequence Neural Networks*. arXiv:1511.05493 [cs, stat]. Sept. 2017.

[19] Michael Edwards and Xianghua Xie. *Graph Based Convolutional Neural Network*. arXiv:1609.08965 [cs]. Sept. 2016.

[20] Armando Vieira. *Knowledge Representation in Graphs using Convolutional Neural Networks*. arXiv:1612.02255 [cs]. Dec. 2016.

[21] Guoqiang Zhang and W. Bastiaan Kleijn. *Training Deep Neural Networks via Optimization Over Graphs*. arXiv:1702.03380 [cs]. June 2017.

[22] Da Xu et al. *Inductive Representation Learning on Temporal Graphs*. arXiv:2002.07962 [cs, stat]. Feb. 2020.

[23] Xiangnan He et al. *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*. arXiv:2002.02126 [cs]. July 2020.

[24] Kelong Mao et al. "UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation". en. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. Virtual Event Queensland Australia: ACM, Oct. 2021, pp. 1253–1262. ISBN: 978-1-4503-8446-9.

[25] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Aug. 2023.

[26] Jie Zou et al. "Improving Conversational Recommender Systems via Transformer-based Sequential Modelling". en. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Madrid Spain: ACM, July 2022, pp. 2319–2324. ISBN: 978-1-4503-8732-3.

[27] Yinwei Wei et al. "LightGT: A Light Graph Transformer for Multimedia Recommendation". en. In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Taipei Taiwan: ACM, July 2023, pp. 1508–1517. ISBN: 978-1-4503-9408-6.

[28] Yuxin Ni et al. "Meta-CRS: A Dynamic Meta-Learning Approach for Effective Conversational Recommender System". en. In: *ACM Transactions on Information Systems* 42.1 (Jan. 2024), pp. 1–27. ISSN: 1046-8188, 1558-2868.

[29] Ziwei Fan et al. "Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer". en. In: *Proceedings of the 30th ACM Inter-*

national Conference on Information & Knowledge Management. Virtual Event Queensland Australia: ACM, Oct. 2021, pp. 433–442. ISBN: 978-1-4503-8446-9.

[30] Qitian Wu et al. "Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems". en. In: *The World Wide Web Conference*. San Francisco CA USA: ACM, May 2019, pp. 2091–2102. ISBN: 978-1-4503-6674-8.

[31] Ehsan Elahi and Zahid Halim. "Graph attention-based collaborative filtering for user-specific recommender system using knowledge graph and deep neural networks". en. In: *Knowledge and Information Systems* 64.9 (Sept. 2022), pp. 2457–2480. ISSN: 0219-1377, 0219-3116.

[32] Cheng Hsu and Cheng-Te Li. *RetaGNN: Relational Temporal Attentive Graph Neural Networks for Holistic Sequential Recommendation*. arXiv:2101.12457 [cs]. Jan. 2021.

[33] Xiaochun Rong. "A Bochner theorem and applications". In: *Duke Mathematical Journal* 91.2 (Feb. 1998). ISSN: 0012-7094.

[34] Steffen Rendle et al. *BPR: Bayesian Personalized Ranking from Implicit Feedback*. arXiv:1205.2618 [cs, stat]. May 2012.

[35] Zhen Yang et al. "Understanding Negative Sampling in Graph Representation Learning". en. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Virtual Event CA USA: ACM, Aug. 2020, pp. 1666–1676. ISBN: 978-1-4503-7998-4.

[36] Yuxin Wang et al. *Efficient Link Prediction via GNN Layers Induced by Negative Sampling*. arXiv:2310.09516 [cs, stat]. Oct. 2023.

[37] Heng-Tze Cheng et al. *Wide & Deep Learning for Recommender Systems*. arXiv:1606.07792 [cs, stat]. June 2016.

[38] Shu Wu et al. "Session-based Recommendation with Graph Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019). arXiv:1811.00855 [cs, stat], pp. 346–353. ISSN: 2374-3468, 2159-5399.

Appendix A

Bochner Theorem

The foundation of our solution for temporal embedding is established upon the principles of classical harmonic analysis, specifically Bochner's theorem. We highlight that our defined temporal kernel K is both positive-semidefinite (PSD) and continuous, as it originates from a Gram matrix and the function Φ is continuous. Thus, the kernel K satisfies the prerequisites of Bochner's theorem, which is articulated as follows.

Theorem 1 (Bochner's Theorem). On $\mathbb{R}^d$, a kernel $K(\mathbf{x}, \mathbf{y}) = \psi(\mathbf{x} - \mathbf{y})$ that is both continuous and translation-invariant is positive definite if there exists a non-negative measure on $\mathbb{R}$ such that ψ is its Fourier transform.

Subsequently, upon appropriate scaling, our temporal kernel K can be represented alternatively as:

$$K(t_1, t_2) = \int e^{i\omega(t_1 - t_2)} p(\omega) d\omega = \mathbb{E}_\omega \left[\cos(\omega(t_1 - t_2))\right], \quad (3)$$

where $\omega(t) = e^{i\omega t}$. Considering that both the kernel K and the probability measure $p(\omega)$ are real, we can derive the real part of equation (3) to attain:

$$K(t_1, t_2) = \mathbb{E}_\omega \left[\cos(\omega t_1)\cos(\omega t_2) + \sin(\omega t_1)\sin(\omega t_2)\right]. \quad (4)$$

The proposed method approximates the expectation via the Monte Carlo integral method,

as suggested by [33]. Hence, $K(t_1, t_2)$ can be approximated by:

$$\frac{1}{d}\sum_{i=1}^{d}\cos(\omega_i t_1)\cos(\omega_i t_2) + \sin(\omega_i t_1)\sin(\omega_i t_2) \tag{A.1}$$

with each ω_i being an i.i.d. sample from $p(\omega)$. We then define a finite-dimensional functional mapping to $\mathbb{R}^d$ as:

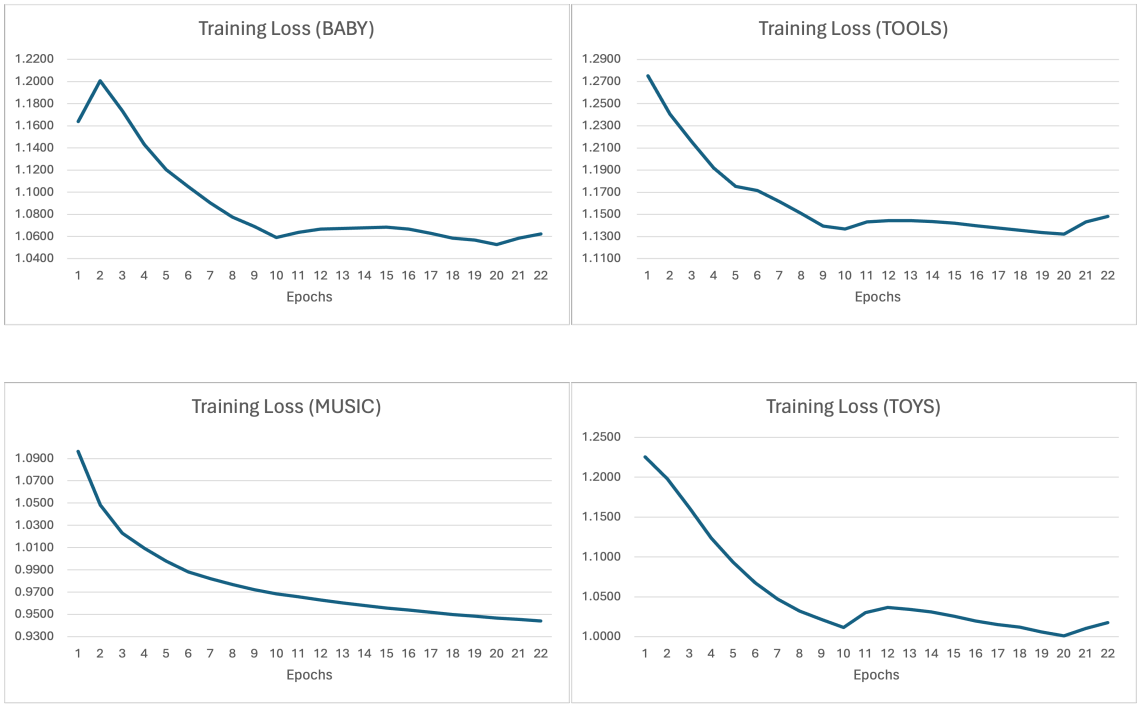
$$t \rightarrow \Phi_d(t) := \sqrt{\frac{1}{d}}\left[\cos(\omega_1 t), \sin(\omega_1 t), ..., \cos(\omega_d t), \sin(\omega_d t)\right], \quad (5)$$

demonstrating that $\langle \Phi_d(t_1), \Phi_d(t_2) \rangle$ closely approximates $K(t_1, t_2)$.

Appendix B

Training Graphs

Figure B.1: BPR Loss variation across Train set during training



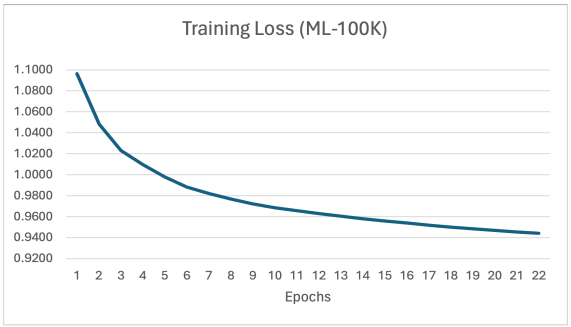


Figure B.2: Average Precision variation across Train and Validation sets during training

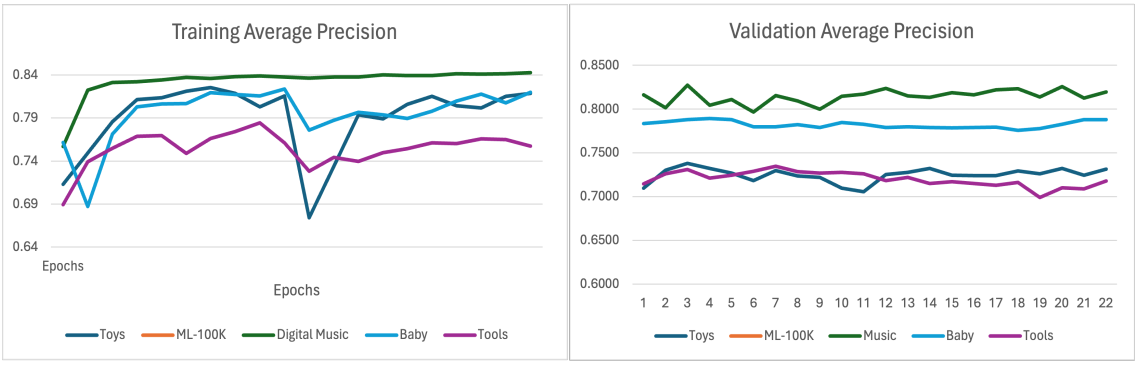
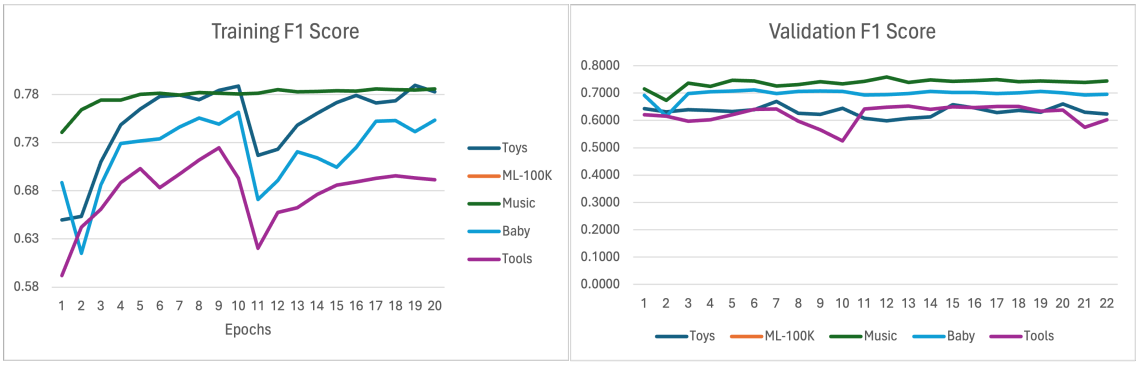



Figure B.3: F1 Score variation across Train and Validation sets during training