PTSR: Prefix-Target Graph-based Sequential Recommendation

Jiayu Chen
Nanjing University of Science and Technology
Nanjing, China
tosakrin@njust.edu.cn

Xiaoyu Du*
Nanjing University of Science and Technology
Nanjing, China
duxy@njust.edu.cn

Yonghua Pan
Nanjing University of Science and Technology
Nanjing, China
yonghuapan@njust.edu.cn

Jinhui Tang
Nanjing University of Science and Technology
Nanjing, China
jinhuitang@njust.edu.cn

ABSTRACT

Sequential recommendation approaches predict the next items (targets) by analyzing prefix subsequences. These methods primarily model the correlations between prefixes and targets but often neglect the inherent correlations among prefixes and items. In this paper, we propose a **P**refix-**T**arget Graph-based **S**equential **R**ecommendation Approach (PTSR), which constructs a prefix-target graph (PTG) to collect observed correlations among prefixes and targets. It utilizes a graph neural network to model these inherent correlations, thus improving the item representations used in the predictive model. Specifically, prefixes linked to the same target reflect similar intents, while targets linked to the same prefix indicate available choices. This allows the graph neural network to effectively capture high-level correlations among prefixes and items, enhancing recommendation accuracy. We conduct extensive experiments on four real-world datasets to demonstrate the superiority of PTSR compared to state-of-the-art (SOTA) sequential recommendation methods. The source code of the PTSR is available at https://github.com/TosakRin/PTSR.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Sequential Recommendation, Graph Neural Networks, Prefix-Target Graph

ACM Reference Format:
Jiayu Chen, Xiaoyu Du, Yonghua Pan, and Jinhui Tang. 2024. PTSR: Prefix-Target Graph-based Sequential Recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3627673.3679718

*Xiaoyu Du is the corresponding author.

1 INTRODUCTION

Sequential recommendation (SR) is a significant branch within the field of recommender systems. It predicts the next possible item a user might be interested in by analyzing the user behavior history. Different from the traditional collaborative filtering methods that rely solely on observing matrices, sequential recommendation approaches incorporate temporal information, where the overall sequences reflect user preferences and the latest subsequences indicate the short-term intents. This provides a more precise understanding of the user's current behavioral intentions, thus offering more accurate and personalized recommendations.

Sequential recommendation approaches treat the observed user interaction sequence as a combination of known prefix subsequence (prefix) and unknown next item (target), aiming to predict the target corresponding to the prefix. Some approaches employ sequence models, e.g., RNN or Transformer [38], to generate the user intent through the prefix. SASRec [16] and BERT4Rec [33] utilize self-attention mechanisms to weight each item in the prefix sequence, better capturing the evolution of user interests and making the model more attentive to items that significantly impact the prediction of the target item. Some other approaches exploit the representation of the target item, incorporating multimodal representations of items. For instance, MMSR [15] explores effective feature fusion of different modalities, providing additional contextual information and cross-modal complementary features for the item.

Recent approaches note that the core of sequential recommendation lies in two inherent correlations among the prefixes and targets, as illustrated in Figure 1(a). Prefix-prefix correlations between two prefixes with the same target reflect they have closely correlated user intents. DuoRec [29] and ICSRec [28] devise contrastive learning over the prefix-prefix correlations to differentiate the user intent at a fine-grained level. Target-target correlations between two targets with the same prefix reveal two available choices for the same user intent. IHGCN [44] builds an item adjacency matrix to present the target-target correlations. Although these methods realize the significance of the correlations, they lack an effective mechanism to incorporate both correlations concurrently.

To address this issue, we propose a **P**refix-**T**arget Graph-based **S**equential **R**ecommendation Approach (PTSR), which incorporates the two inherent correlations concurrently to improve the sequential recommendation. Specifically, PTSR constructs a prefix-target graph (PTG), where the nodes indicate the prefixes and items, and

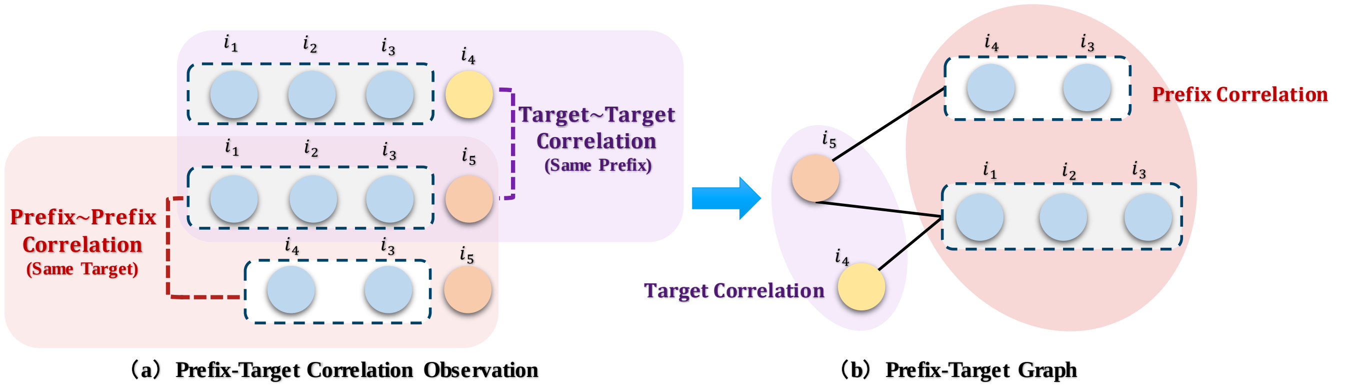

Figure 1: (a) An example of two latent relationships between prefix sequences and target items. The red block describes the prefix-prefix correlation that different prefix sequences are linked to the same target item. The purple part denotes the target-target correlation that different target items have the same prefix sequence. (b) The Prefix-Target Graph (PTG) with the connections of the natural prefix-target transitions. The inherent target-target and prefix-prefix correlations are two-hop correlations in this Prefix-Target Graph.

the edges indicate the prefix-target transition existing in the observation history. As shown in Figure 1(b), with just one-stage message propagation along the PTG, PTSR can incorporate the inherent prefix-prefix and target-target correlations into representation learning. Therefore, PTSR applies a graph neural network (GNN) over PTG to enhance the original item representations, thus improving the results of sequential recommendations. We conduct extensive experiments on four datasets. The results demonstrate the effectiveness of PTG.

To summarize, our work makes the following contributions:

- We propose a novel prefix-target graph-based sequential model, PTSR, which effectively captures the transition correlations between prefixes and targets to enhance the representations of items in sequential recommendations.
- We introduce the method for constructing a Prefix-Target graph and fully explore the impact of the inherent prefix-prefix and target-target correlations.
- We conduct extensive experiments on four real-world datasets, demonstrating that PTSR outperforms state-of-the-art sequential recommendation approaches.

2 RELATED WORK

2.1 Sequential Recommendation

Sequential recommendation captures the user intents by analyzing their ordered interaction sequences [30]. The core lies in predicting the next item a user will likely interact with based on their past interaction history.

The canonical methods for sequential recommendation are dedicated to modeling the sequential patterns in user-item interactions. For example, FPMC [32] uses Markov chain models [10, 27] to capture the sequential patterns in user sequences. GRU4Rec [13] and Caser [36] apply recurrent neural networks (RNN) and convolutional neural networks (CNN) to model the sequential interactions. The Transformer [38] and attention mechanisms have been introduced into the sequential recommendation [21] to capture the complex short and long-term dependencies in user sequences. SASRec [16], BERT4Rec [33], and SSE-PT [41] are representative models that leverage the self-attention mechanism to capture the temporal dependencies between user interactions with items, which weight each item in the prefix sequence to model the user's interest evolution process better.

Although numerous users and items exist on platforms, most users/items are only associated with a few interactions, resulting in poor generalization and prediction capabilities. While early works can not solve the sparsity problem in user-item interactions, recent methods [22] have attempted to address this issue by leveraging self-supervised learning techniques, such as contrastive learning [8]. CL4SRec [43] is the first to introduce contrastive learning to the sequential recommendation, using crop, mask, and shuffle operations to augment the input sequence and maximize the mutual information between positive and negative samples. S^3-Rec [46] uses four sub-task objectives to learn the representations of items and apply InfoNCE [26] loss to realize maximum mutual information. CoSeRec [23] introduces additional substitute and insert enhancement methods based on CL4SRec. DuoRec [29] abandons traditional data augmentation methods that would corrupt the original sequence and it utilizes the randomness of dropout in the model to form different views for unsupervised contrastive learning.

The above methods mainly focus on item-level transitions in user sequences, which can be considered item-aware methods. More recently, the viewpoint that the user intent is more important than just the item itself in sequential recommendation has received increasing attention [2]. Many intent-aware methods propose to capture the user's intention transition in the sequence. DSSRec [24] proposes a seq2seq model and introduces a variable representing user intent. ICLRec [3] builds the intent prototype by clustering the user's historical behavior sequence representations. IOCRec [20] disentangles the latent intents from the user behavior sequence and uses the intent to guide the item representation learning. ICSRec [28] extracts subsequences as multiple intents from the user behavior sequence and constructs positive pairs based on the target item of the user's subsequences.

Although these models have attempted to better capture user intention transitions by modeling the relationships between prefix sequences and target items, they still overlook the higher-order relationships between target items and prefix sequences, which may provide more comprehensive information for capturing user intention transitions.

2.2 Graph-based Recommendation

Most of the information in recommendation systems essentially has a graph structure, which aligns well with the strengths of graph neural networks (GNNs) [18] in graph representation learning. From the perspective of graph structure, different data types can be modeled using a unified framework. By transmitting information through multi-layer networks, GNN can explicitly encode higher-order signals in user interaction behavior. Consequently, GNN-based methods have been widely used in the recommendation system to model user-item interactions and item-item relationships. NGCF [39] initially brought the GNN method into recommendation systems, which can capture the collaborative filtering signal and the graph structure signal simultaneously. LightGCN [11] simplifies the NGCF model and removes the feature transformation layer, achieving better performance with fewer parameters. PinSage [45] uses a graph convolutional network (GCN) to learn the representation of items in a heterogeneous graph and generates item embeddings by aggregating the embeddings of neighboring items. GraphSAGE [6] proposes a sampling strategy to learn the representation of nodes in the graph, which can be applied to large-scale graphs.

Building on experiences in the general recommendation, graph-based methods have also been applied to sequential recommendations to model the complex relationships between user sequences and items. The SR-GNN [42] introduces graph-based structures to model session sequences by representing sessions as directed graphs. GCE-GNN [40] leverages contrastive learning within a GNN framework to enhance node representation learning by maximizing agreement between different augmented views of the same graph. SURGE [1] models user-item interactions and item-item relationships in a unified graph structure, generating and leveraging augmented graph views to learn robust node representations without requiring labeled data. IHGCN [44] applies GCN to capture the similarity of items that share the same prefix sequence, enriching the representations of items through GCN layers.

Although these methods have proposed various graph-building strategies, they still use graphs to build item-level relationships rather than intent-level relationships, struggling to capture the user's intent in the recommendation process.

3 METHOD

This section formally defines the sequential recommendation task and introduces the proposed PTSR. We first extract subsequences from the original user sequences to build prefix sequences and target item pairs. Next, we construct the Prefix-Target Graph (PTG) to capture distinctive representations of items with GNN aggregating the transition and correlation information between pairs. Given the enhanced item representations, we predict the next item by a self-attention recommender. The overall architecture of the PTSR is illustrated in Figure 2.

3.1 Problem Formulation

For the sequential recommendation, let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ and $\mathcal{I} = \{i_1, i_2, \ldots, i_{|\mathcal{I}|}\}$ denote the set of users and items, respectively, where $|\mathcal{U}|$ and $|\mathcal{I}|$ is the number of unique users and items.

For each user $u \in \mathcal{U}$, there's an interaction history sequence $s^u = [i_1^u, i_2^u, \ldots, i_T^u]$ arranged chronologically, where T denotes the length of that sequence. Each $i_n^u \in \mathcal{I}, 0 \leq n \leq T$ in the interaction history refers to the item that u interacted with at time step n.

Based on the preferences implied by item transitions in users' behavior history, the sequential recommendation aims to forecast the next item for a given user interaction sequence s^u. Mathematically, the next-item prediction can be formulated as:

$$\hat{i} = \arg\max_{i \in \mathcal{I}} p(i|s^u), \tag{1}$$

where $\hat{i}$ is the item predicted after observing s^u, and $p(i|s^u)$ is the probability of item i as the next interaction closely following s^u.

3.2 Prefix-Target Graph Construction

The prefix sequences and target items within sequences can offer abundant transition and correlation information. The direct prefix-target connections suggest temporal transitions between prefix sequences and target items, while the high-order connections about prefixes pointing to an identical target and targets sharing the same prefix imply the correlation between them. We construct the Prefix-Target Graph (PTG) To build these relationships concurrently.

3.2.1 ***Prefix-Target Pair Extraction.*** Initially, we extract subsequences from all original user sequences to leverage the intrinsic relationships between prefix sequences and target items. Building on prior research [28, 35], we split the complete user sequence into multiple subsequences using a dynamic sliding operation. For each user behavior sequence $s^u = [i_1^u, i_2^u, \ldots, i_T^u]$, the dynamic sliding window begins by setting its start index $I_s = 0$. It incrementally moves the end index I_e from a minimal subsequence length l_{min}. This process continues until the end index I_e either reaches the max length of the complete sequence T or the predefined maximum length l_{max}. If the end index I_e has already increased to l_{max} but still hasn't yet reached the tail of the complete sequence, the window starts sliding step-by-step, with I_s and I_e incrementing by one each time, maintaining the window length at T. The process is repeated until the end of the sequence. All subsequences encompassed by the window during the dynamic sliding are collected, forming the subsequence set $\mathcal{S}^u = \{s_1^u, s_2^u, \ldots, s_k^u\}$ belonging to u, where k is the total number of subsequences for s^u. All subsequences from every user are denoted as the subsequence set $\mathcal{S} = \bigcup_{u \in \mathcal{U}} \mathcal{S}^u$. This extraction secures our ability to capture various levels of user intents and transitions within the interaction history, providing a more comprehensive representation of user behavior patterns.

To build the pair of a target item with corresponding prefix sequence for any given subsequence $s = [i_1, i_2, \ldots, i_{|s|}]$ in the PTG construction and also the recommendation training, we use the last item of the subsequence as the target item and the remaining items before the last one as the prefix sequence, which is formally

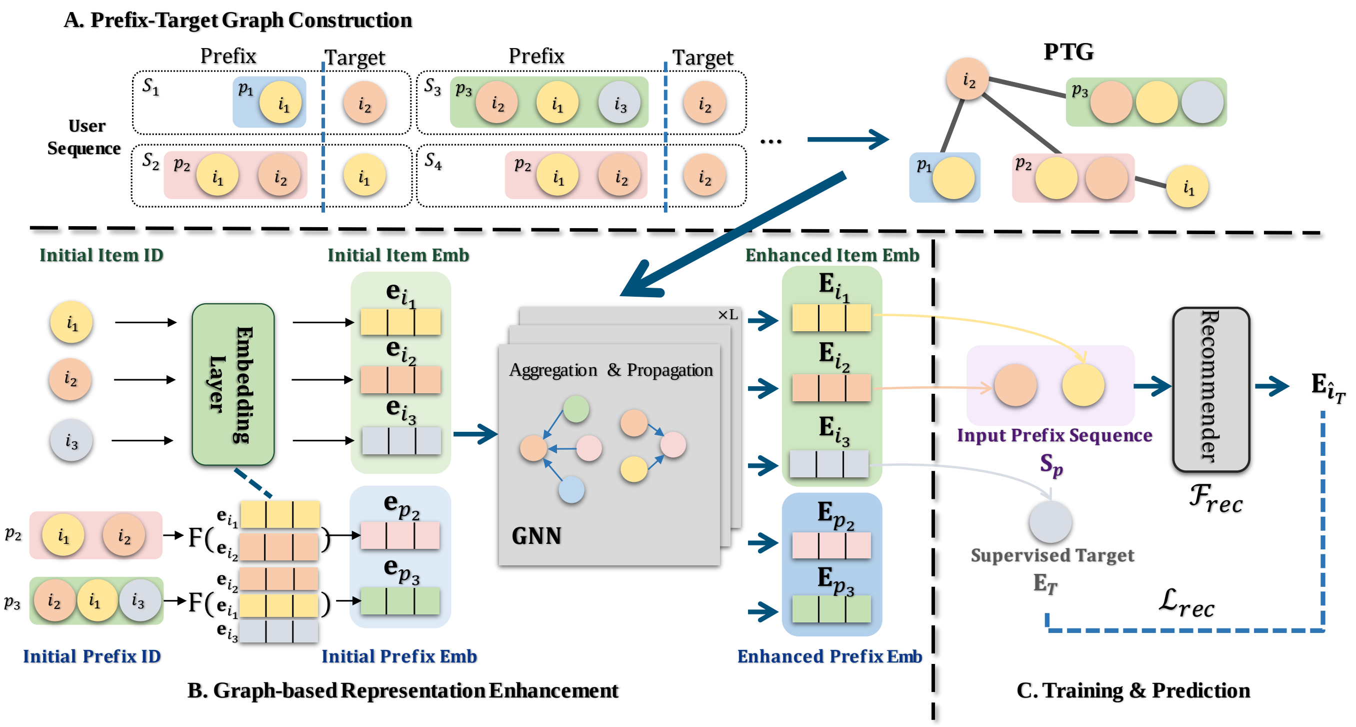

Figure 2: Overview of PTSR. (A) The construction of Prefix-Target Graph (PTG) from the user sequences with paired prefixes and targets. (B) The representation enhancement with the inherent Prefix-Prefix and Target-Target correlations by employing GNN over PTG. (C) Sequential Recommendation with the enhanced representations.

denoted as:

$$\begin{aligned} \text{Prefix:} \quad & p = [i_1, i_2, \ldots, i_{|s|-1}], \\ \text{Target:} \quad & t = i_{|s|}. \end{aligned} \tag{2}$$

To construct the global PTG containing all correlations between prefix sequences and target items, we collect prefix sequences and target items from all users' subsequences. It eventually forms a set of prefix-target pairs, denoted as:

$$\mathcal{D} = \{(p, t) | s \in \mathcal{S}\}. \tag{3}$$

3.2.2 ***Prefix-Target Graph Construction***. When two prefix sequences have the same target item, the user intent described by the sequence is correlated. Similarly, when the same sequence points to two different targets, both targets are feasible options. Having prefix-target pairs, we arrange them into a graph form to enable the later algorithm to use the relationship conveniently. Specifically, to capture these transition relationships between prefix sequences and target items, as well as prefix-prefix and target-target correlations, we construct the PTG.

We first create a graph structure whose nodes represent prefix sequences and target items, and edges represent relationships between prefixes and their corresponding targets. Iterating through the set of prefix sequences and target item pairs $\mathcal{D}$, we add nodes for each unique prefix sequence and target item to the graph. We form the graph's edges by connecting each prefix sequence node to its corresponding target item node.

Formally, the PTG can be represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ is the set of nodes representing prefix sequences and target items, $\mathcal{E}$ is the set of edges connecting each prefix sequence node to its corresponding target item node. The adjacency matrix $\mathbf{A}$ represents the connections between nodes in the graph. Specifically, $\mathbf{A}$ is a sparse matrix with dimensions $|\mathcal{V}| \times |\mathcal{V}|$, where $\mathbf{A}_{ij} = 1$ if there is an edge between nodes i and j, and $\mathbf{A}_{ij} = 0$ otherwise. The complete Prefix-Target Pair Extraction and Prefix-Target Graph Construction procedure are shown in Algorithm 1.

The PTG using graph-based data organization allows us to easily model transition and correlation information. Intuitively, the one-hop neighbors are the corresponding prefix sequences for each target item node $t \in \mathcal{V}$. In the same way, the one-hop neighbors of a prefix sequence node $p \in \mathcal{V}$ are the target item connected to it. The one-hop neighbors capture the immediate transition between prefix sequences and target items. By exploiting the one-hop neighbor, we can learn about the transition relationship. In contrast, the two-hop neighbors of the target item point to target items with the same prefix sequence, and the two-hop neighbors of the prefix sequence turn back to the prefix sequences with the same target item. Based on this higher-order signal correlation, collecting the two-hop neighbor information allows us to learn the collaborative correlation between similar target items and prefix sequences. By incorporating the one-hop and two-hop neighbor information, we can integrate the relationship of the Prefix-Target transition and build a connection between the prefix-prefix relation and target-target correlation in the graph simultaneously. Moreover, we can

Algorithm 1 Prefix-Target Graph Construction

```
Require: Set of users $\mathcal{U}$, Set of items $\mathcal{I}$, User interaction sequences $\{s^u | u \in \mathcal{U}\}$, Minimum subsequence length $l_{min}$, Maximum subsequence length $l_{max}$
Ensure: Prefix-Target Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$
 1: $\mathcal{D} \leftarrow \emptyset$
 2: for each user $u \in \mathcal{U}$ do
 3:     $s^u \leftarrow$ interaction sequence of user $u$
 4:     for $I_s \leftarrow 0$ to $T - 1$ do
 5:         for $I_e \leftarrow I_s + l_{min}$ to $\min(I_s + l_{max}, T)$ do
 6:             $p \leftarrow [i^u_{I_s}, \ldots, i^u_{I_e - 2}]$
 7:             $t \leftarrow i^u_{I_e - 1}$
 8:             $\mathcal{D} \leftarrow \mathcal{D} \cup \{(p, t)\}$
 9:         end for
10:     end for
11: end for
12: $\mathcal{V} \leftarrow \{v \mid (p, t) \in \mathcal{D}, v \in \{p, t\}\}$
13: $\mathcal{E} \leftarrow \mathcal{D}$
14: $\mathbf{A} \leftarrow \mathbf{0}_{|\mathcal{V}| \times |\mathcal{V}|}$
15: for each $(p, t) \in \mathcal{E}$ do
16:     $\mathbf{A}[p, t] \leftarrow 1$
17:     $\mathbf{A}[t, p] \leftarrow 1$
18: end for
19: return $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$
```

further extend the neighbor to k-hop to capture more high-order interactions between nodes by leveraging the interaction path implied by adjacency matrix $\mathbf{A}$.

3.3 Graph-based Representation Enhancement

With the PTG available, we can use GNN to integrate the transition and correlation relationships between target items and prefix sequences into their representation.

3.3.1 ***Node Representation.*** In the PTG, each node is represented by a dense vector, which is the embedding of the target item or prefix sequence. We embed all the items in the dataset into a low-dimensional space using an embedding layer. The embedding of item i can be denoted as $\mathbf{e}_i \in \mathbb{R}^d$. Since prefix sequences are lists of items, we can fully utilize the inner relationship between them. We directly represent a prefix sequence by aggregating the representation of items within it.

Specifically, we treat all the items in a prefix sequence equally and use the average pooling operation to aggregate all the items from the prefix sequence into a single embedding vector. Formally, the embedding of prefix sequence p can be expressed as:

$$\mathbf{e}_p = \frac{1}{|p|} \sum_{i \in p} \mathbf{e}_i, \tag{4}$$

where $\mathbf{e}_i$ represents the embedding of item i, and $|p|$ denotes the length of the prefix sequence p.

3.3.2 ***GNN Layer.*** The above process combines all the dataset's prefix sequences and target items to form a global graph. With the PTG $\mathcal{G}$ constructed, we can leverage GNN to learn the representations of prefix sequences and target items by fully exploiting the transition and correlation information in PTG.

GNN aggregates information from a node's neighbors to update its representation. Due to the lack of semantic information in ID-only scenarios, we adopt LightGCN [11] as our backbone. This simplified GNN allows for more efficient learning of node embeddings by directly leveraging the graph structure without requiring feature transformation or nonlinear activation functions. For given prefix sequence p and target item i, the update rule of LightGCN can be formulated as:

$$\mathbf{e}_p^{(l+1)} = \sum_{i \in \mathcal{N}_p} \frac{1}{\sqrt{|\mathcal{N}_p|\,|\mathcal{N}_i|}} \mathbf{e}_i^{(l)}, \quad \mathbf{e}_i^{(l+1)} = \sum_{p \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|\,|\mathcal{N}_p|}} \mathbf{e}_p^{(l)}, \tag{5}$$

where $\frac{1}{\sqrt{|\mathcal{N}_p||\mathcal{N}_i|}}$ is the symmetrically normalized term of the graph adjacency matrix $\mathbf{A}$. $\mathcal{N}_p$ and $\mathcal{N}_i$ denote the one-hop neighbors of prefix sequence s and target item i.

3.3.3 ***Enhanced Node Representation.*** To further enhance our model's representation capability and catch high-order interactions, we stack multiple GNN layers to learn the multi-layered representation of items and prefix sequences. The final representation of the graph-enhanced embedding is obtained by combining all the layers' embeddings, which can be formulated as:

$$\mathbf{E}_i = \sum_{l=0}^{L} \alpha_l \cdot \mathbf{e}_i^{(l)}, \quad \mathbf{E}_p = \sum_{l=0}^{L} \alpha_l \cdot \mathbf{e}_p^{(l)}, \tag{6}$$

where α_l is the learnable weight for the l-th layer, and $\mathbf{e}_i^{(l)}$ and $\mathbf{e}_p^{(l)}$ are the embeddings of target item i and prefix sequence p in the l-th layer, respectively. To avoid unnecessary complexity, we set α_l uniformly as $1/(L+1)$, which means each layer contributes equally to the enhanced node representation, ensuring transition and correlation relationships have a balanced contribution to representation.

We finally obtain the enhanced representations of items and prefix sequences for the sequential recommendation task. The prefix sequence embeddings get updated every epoch while the enhanced item embeddings learn their representation following the GCN propagation every batch. Although we do enhance the prefix sequence embeddings, we only use them to enrich the target item embeddings in GNN. The prefix sequence embeddings can't be directly used as sequential recommender input since prefix sequences are dynamic and fail to cover unseen variant user histories. So, we use the universal item embeddings to represent the prefix sequence as recommendation task input.

To sum up, the usage of PTG is for enhancing the item embeddings by integrating the sequence information based on the correlation between prefixes and targets. The prefix sequence embeddings get updated every epoch while the enhanced item embeddings learn their representation following the GCN propagation every batch. However, when prefixes serve as input for the recommendation task, we do not represent them directly with their PTG-enhanced embeddings but with their items, using the enhanced items embeddings for representation.

Table 1: Detailed statistics of experimental datasets

Statistics	**Beauty**	**Sports**	**Toys**	**ML-1M**
# Users	22,363	35,598	19,412	6,040
# Items	12,101	18,357	11,924	3,416
# Interactions	198,502	296,337	167,597	999,611
Avg. Actions/User	8.8	8.3	8.6	165.4
Avg. Actions/Item	16.4	16.1	14	292.6
Sparsity	99.93%	99.95%	99.93%	95.16%

3.4 Training & Prediction

It is worth noting that the enhanced representation incorporates transition patterns, prefix-prefix correlations, and target-target relevance. With graph-based representation enhancement, we can insert enhanced item representation into various sequential recommendation backbones to realize the next-item prediction. In this work, we adopt the self-attention-based sequential recommendation model SASRec as our sequential recommender.

During the training stage, the prefix-target pairs from extracted subsequences are used as the user behavior sequence and ground truth next item respectively. As representations of user intents, the prefix sequences from subsequences can offer more fine-grained information about user behavior patterns, which can help the model capture the user's intention transition better. So we augment the training data by generating multiple training samples from a single complete user sequence. It makes the model more capable of generalizing better to different sequence lengths and patterns.

To capture the short and long-term dependencies in the user behavior sequences and model the interactions between items in the sequences, we use the enhanced item embeddings directly as the input for the sequential recommender, replacing the original item embeddings in the user sequence. For the subsequence $s = [i_1, i_2, \cdots, i_T]$, the input for the training recommender can be formulated as follows:

$$\mathbf{S}_p = [\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_{T-1}], \tag{7}$$

and ground truth target item t is $\mathbf{E}_T$. We feed the enhanced embeddings of the prefix sequence into the recommender $\mathcal{F}_{rec}$ to get the prediction representation of the next item $\hat{i}$ as follows:

$$\mathbf{E}_{\hat{i}} = \mathcal{F}_{rec}(\mathbf{S}_p). \tag{8}$$

The recommender will output the final prediction representation $\mathbf{E}_{\hat{i}}$ through forward propagation. We predict the next item by calculating the probability of each item in the item set $\mathcal{I}$ with $\mathbf{E}_{\hat{i}}$:

$$\hat{\mathbf{p}} = \text{Softmax}(\mathbf{E}_{\hat{i}} \cdot \mathbf{E}_i | i \in \mathcal{I}) \tag{9}$$

where $\hat{\mathbf{p}}$ is the predicted probability distribution over all items in the item set $\mathcal{I}$, $\mathbf{E}_i$ is the graph-based enhanced embedding of item i, and $(\cdot)$ is dot-product operation. By ordering, we can obtain the top-N suggested outcomes for predicting the subsequent item.

We use the cross-entropy loss function to compute the optimized objective for the main supervised recommendation task. The final loss function for the recommendation task $\mathcal{L}$ is given by:

$$\mathcal{L}_{rec} = -\sum_{(p,t) \in \mathcal{D}} \log \hat{\mathbf{p}}(t). \tag{10}$$

4 EXPERIMENTS

In this section, we conduct extensive experiments attempting to answer the following research questions (**RQs**):

- **RQ1**: How does PTSR perform compared with the state-of-the-art methods?
- **RQ2**: Does PTSR benefit from the core modules prefix sequence, PTG, and GNN?
- **RQ3**: How do the hyperparameters, the number of GNN layers, the length of prefix sequences, and the connection numbers in PTG impact the performance of PTSR?

4.1 Experimental Setting

4.1.1 ***Datasets.*** Following previous works [4, 9, 16, 19, 33], we conduct experiments on four public real-world benchmarks using the same data preprocessing methods.

- **Amazon** [25]: A series of datasets that collect user reviews on products from Amazon.com. The dataset can be divided into subsets according to product categories with relatively short sequences. In this work, we pick **Sports**, **Beauty**, and **Toys** as three different experimental datasets.
- **MovieLens** [7]: A widely used recommendation system dataset containing movie user ratings. We use the MovieLens-1M dataset, a subset of the MovieLens dataset with 1 million ratings.

We convert each dataset into an implicit format by treating each rating or review as indicative of a user-item interaction. Following the practices outlined by [12, 16, 37], sequences and items with fewer than five interactions were excluded from the analysis. The statistical properties of the datasets are summarized in Table 1, which illustrates the variation in size and density among the datasets. This diversity enables a comprehensive evaluation of the proposed models across different contexts.

4.1.2 ***Evaluation Metrics.*** To evaluate the performance of each method, we use the widely accepted leave-one-out evaluation task for each sequence. This approach involves reserving the last item in the sequence for testing purposes, the second-to-last item for validation, and the remaining items for training. Moreover, we adopt a full-sort evaluation method that predicts the next item by ranking all the items in the item set based on the predicted probability distribution for a fair comparison. We employ two standard evaluation metrics to assess the ranked list: Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). We report HR and NDCG with Top-k recommendation, where $k = 5, 10, 20$. They yield high values when the ground truth item appears higher on the Top-k list. Given that only one ground truth item exists for each user, HR@k is effectively the same as Recall@k.

4.1.3 ***Baseline Models.*** To demonstrate the effectiveness of our approach, we compare PTSR with the state-of-the-art baseline methods as follows:

Non-Sequential Models.

- **BPR** [31] is the first to employ Bayesian Personalized Ranking (BPR) loss to optimize the matrix factorization model as a representative of non-sequential recommendation,

Conventional Sequential Models.

Table 2: Performance comparisons of different methods. The results of the best baseline are underlined in each row and bolod values denote the best results. The last column is the relative improvements compared with the best baseline results.

Dataset	Metric	BPR	GRU4Rec	Caser	SASRec	BERT4Rec	S^3-Rec$_{MIP}$	CL4SRec	CoSeRec	DuoRec	DSSRec	SINE	ICLRec	IOCRec	ICSRec	**PTSR**	**impro.**
Sports	HR@5	0.0123	0.0162	0.0154	0.0214	0.0217	0.0121	0.0231	0.0290	0.0312	0.0209	0.0240	0.0290	0.0293	0.0403	**0.0432**	6.16%
	HR@10	0.0215	0.0258	0.0261	0.0333	0.0359	0.0205	0.0369	0.0439	0.0466	0.0328	0.0389	0.0437	0.0452	0.0565	**0.0600**	8.23%
	HR@20	0.0369	0.0421	0.0399	0.0500	0.0604	0.0344	0.0557	0.0636	0.0696	0.0499	0.0610	0.0646	0.0684	0.0794	**0.0844**	9.48%
	NDCG@5	0.0076	0.0103	0.0114	0.0144	0.0143	0.0084	0.0146	0.0196	0.0195	0.0139	0.0152	0.0191	0.0169	0.0283	**0.0297**	4.45%
	NDCG@10	0.0105	0.0142	0.0135	0.0177	0.0190	0.0111	0.0191	0.0244	0.0244	0.0178	0.0199	0.0238	0.0220	0.0335	**0.0351**	5.70%
	NDCG@20	0.0144	0.0186	0.0178	0.0218	0.0251	0.0146	0.0238	0.0293	0.0302	0.0221	0.0255	0.0291	0.0279	0.0393	**0.0413**	6.79%
Beauty	HR@5	0.0178	0.0180	0.0251	0.0377	0.0360	0.0189	0.0401	0.0504	0.0561	0.0408	0.0354	0.0500	0.0511	0.0698	**0.0741**	7.20%
	HR@10	0.0296	0.0284	0.0342	0.0624	0.0601	0.0307	0.0642	0.0725	0.0851	0.0616	0.0612	0.0744	0.0774	0.0960	**0.1039**	6.19%
	HR@20	0.0474	0.0478	0.0643	0.0894	0.0984	0.0487	0.0974	0.1034	0.1228	0.0894	0.0963	0.1058	0.1146	0.1298	**0.1421**	6.30%
	NDCG@5	0.0109	0.0116	0.0145	0.0241	0.0216	0.0115	0.0268	0.0339	0.0348	0.0263	0.0213	0.0326	0.0311	0.0494	**0.0516**	4.95%
	NDCG@10	0.0147	0.0150	0.0226	0.0342	0.0300	0.0153	0.0345	0.0410	0.0441	0.0329	0.0296	0.0403	0.0396	0.0579	**0.0612**	4.78%
	NDCG@20	0.0192	0.0186	0.0298	0.0386	0.0391	0.0198	0.0428	0.0487	0.0536	0.0399	0.0384	0.0483	0.0490	0.0663	**0.0708**	5.09%
Toys	HR@5	0.0122	0.0121	0.0205	0.0429	0.0371	0.0456	0.0503	0.0533	0.0655	0.0447	0.0385	0.0597	0.0542	0.0788	**0.0820**	4.06%
	HR@10	0.0197	0.0184	0.0333	0.0652	0.0524	0.0689	0.0736	0.0755	0.0959	0.0671	0.0631	0.0834	0.0804	0.1055	**0.1096**	3.89%
	HR@20	0.0327	0.0290	0.0542	0.0957	0.0760	0.0940	0.0990	0.1037	0.1293	0.0942	0.0957	0.1139	0.1132	0.1368	**0.1452**	6.14%
	NDCG@5	0.0076	0.0077	0.0125	0.0245	0.0259	0.0314	0.0264	0.0370	0.0392	0.0297	0.0225	0.0404	0.0297	0.0571	**0.0592**	3.68%
	NDCG@10	0.0100	0.0097	0.0168	0.0320	0.0309	0.0388	0.0339	0.0442	0.0490	0.0369	0.0304	0.0480	0.0381	0.0657	**0.0681**	3.65%
	NDCG@20	0.0132	0.0123	0.0221	0.0397	0.0368	0.0452	0.0404	0.0513	0.0574	0.0437	0.0386	0.0557	0.0464	0.0736	**0.0771**	4.76%
ML-1M	HR@5	0.0247	0.0806	0.0912	0.1078	0.1308	0.1078	0.1142	0.1128	0.2098	0.1371	0.0990	0.1382	0.1796	0.2445	**0.2515**	2.86%
	HR@10	0.0412	0.1344	0.1442	0.1810	0.2219	0.1952	0.1815	0.1861	0.3078	0.2243	0.1694	0.2273	0.2689	0.3368	**0.3483**	3.41%
	HR@20	0.0750	0.2081	0.2228	0.2745	0.3354	0.3114	0.2818	0.2950	0.4098	0.3275	0.2705	0.3368	0.3831	0.4518	**0.4579**	1.35%
	NDCG@5	0.0159	0.0475	0.0565	0.0681	0.0804	0.0616	0.0705	0.0692	0.1433	0.0898	0.0586	0.0889	0.1201	0.1710	**0.1742**	1.87%
	NDCG@10	0.0212	0.0649	0.0734	0.0948	0.1097	0.0917	0.0920	0.0915	0.1749	0.1179	0.0812	0.1175	0.1487	0.2007	**0.2054**	2.34%
	NDCG@20	0.0297	0.0834	0.0931	0.1156	0.1384	0.1204	0.1170	0.1247	0.2007	0.1440	0.1066	0.1450	0.1775	0.2297	**0.2330**	1.44%

- **GRU4Rec** [14] is the pioneering work of deep learning, introducing a Gated Recurrent Unit (GRU) to model sequences and first leveraging RNN for SR.
- **Caser** [36] first introduces CNN to SR, which leverages both horizontal and vertical convolution to model the sequence.
- **SASRec** [16], introducing the Transformer-based framework, first utilizes the attention mechanism to model sequences, which greatly improves the performance of SR.
- **BERT4Rec** [33] first introduces BERT [5] to model the sequence, which employs a bidirectional attentive encoder and leverages the Mask Item Predict (MIP) task to capture the potential relationships between items and sequences.

Item-aware Sequential Models.

- **S^3-Rec$_{MIP}$** [46] proposes four types of learning objectives and applies MIP to realize maximum mutual information. Since we have no attribute information, only the MIP task, called S^3-Rec$_{MIP}$, is used for training.
- **CL4SRec** [43] combines a multi-task contrastive learning model with a multi-head self-attention mechanism, and three stochastic enhancement operators are proposed for generating positive and negative samples in contrastive learning.
- **CoSeRec** [23] is a contrastive learning framework based on the Transformer encoder that proposes two informative robust data enhancement operators based on CL4SRec.

Intent-aware Sequential Models.

- **DuoRec** [29] utilizes the randomness of dropout in the model to form different views for unsupervised contrastive learning and uses contrastive learning on sequences with the same next item.
- **DSSRec** [24] proposes a seq2seq model and introduces a variable representing user intent.
- **SINE** [34] designs a concept pool and retrieves user intent by behavior sequence in the concept pool.
- **ICLRec** [3] builds the intent prototype by clustering the user historical behavior sequence representations.
- **IOCRec** [20] disentangles the latent intents from the user behavior sequence and uses the intent to guide the item representation learning.
- **ICSRec** [28] uses subsequences as training data and applies contrastive learning between similar subsequences to enhance the representation.

4.1.4 ***Implementation Details***. We adopt the original code with the original settings for the baselines to reproduce the results on four benchmarks. We employ the Adam [17] optimizer with the batch size set to 256. The model has 2 Transformer layers, 4 GNN layers, and a 64-bit latent embedding size. The number of self-attention blocks and attention heads is set to 2. We set d as 64 and T as 50. The PTG augmentation is performed once per batch. While

Table 3: The HR@5 and NDCG@5 performances achieved by ICSRec variants and SASRec on four datasets.

Model	Dataset					
	Sports		Beauty		Toys	
	HR@5	NDCG@5	HR@5	NDCG@5	HR@5	NDCG@5
(A) PTSR	**0.0432**	**0.0297**	**0.0741**	**0.0516**	**0.0820**	**0.0592**
(B) w/o GCN	0.0408	0.0282	0.070	0.0508	0.0774	0.0564
(C) w/o Subseq	0.0268	0.0188	0.0487	0.0339	0.0511	0.0372
(D) SASRec	0.0214	0.0144	0.0377	0.0241	0.0429	0.0245

in the testing stage, we use original complete user sequences to predict the next item instead of subsequences.

4.2 Overall Performance (RQ1)

We compare the performance of all baselines with the PTSR. Table 2 shows the experimental results of all the compared models on four datasets. Accordingly, we have the following observations,

- Benefiting from introducing prefix sequences and Prefix-Target Graph, PTSR significantly outperforms other methods on all metrics across the different datasets. Specifically, PTSR improves the best baselines by 9.48%, 6.03%, 6.14%, and 1.35% regarding HR@20 on Beauty, Sports, Toys, and ML-1M, respectively. Looking at all six key metrics, the model improves by an average of 6.80%, 5.75%, 4.36%, and 2.21% across the four datasets compared with the SOTA baseline ICSRec.
- The non-sequential result, BPR, can hardly achieve a comparable result with other sequential methods since such a shallow method lacks the representation learning capability of users' historical behaviors. The first representative deep learning method is GRU4Rec, which can consistently outperform the non-sequential BPR. It can be concluded that the incorporation of sequential information can improve performance.
- Attention-based models, e.g., SASRec and BERT4Rec, can perform better than CNN and RNN-based models since the attention mechanism makes the model pay more attention to the items that have a significant impact on the current target item prediction. Based on attention models, CL4SRec and CoSeRec introduce discriminative representations by contrastive learning from an item-aware perspective, achieving significant progress in performance.
- Intent-aware methods that further mine user intents implied by user sequences attain the top results in our experiments. They are designed to capture user intents among item transitions, which can more accurately reflect users' real preferences compared to item-aware methods that only consider item-level transition relationships. Similarly, as an intent-aware method, PTSR leverages low-order connections like prefix-target transitions while simultaneously capturing high-order relationships such as prefix-prefix and target-target correlations, thereby achieving state-of-the-art performance.

4.3 Ablation Study (RQ2)

We perform ablation studies developing three variants, with each one excluding a specific key component, to delve deeper into the design of PTSR.

- **w/o GNN** removes the GNN graph learning that enhances the item representation by incorporating prefix sequences.
- **w/o Subseq** removes the usage of prefix sequences during the training stage, implying that the original user sequences are used.
- **w/o both** removes the entire PTSR. This variant functions equivalently to SASRec.

The results of the ablation study are presented in Table 3. In the ablation study, we can make the following observations: 1) Each of these critical components contributes substantially to the enhancement of the model's recommendation performance; 2) After using prefix sequences as training, we can significantly improve the model's performance thanks to the introduction of the prefix-sequence intent signal; 3) We obtained the SOTA model after further using the GNN to aggregate interests around items.

4.4 Further Discussion (RQ3)

We conduct further discussions to investigate some details of the PTSR.

4.4.1 ***Impact of Graph Layers.*** We modify the graph layer number of PTSR from 0 to 6, and conduct experiments on the three datasets to investigate the impact of the layer number on performance. The results are shown in Figure 3. We can observe that when the layer number is 0, that is, when the GNN module is removed, the model's performance degrades significantly. Once the Prefix-Target GNN layer is introduced, the model's performance increases significantly, even if only one layer is added. According to a rough comparison, the performance of GNN-based models with one layer is often the lowest, and model performance grows slightly with the number of layers. However, the layer number generally doesn't significantly impact the model's performance. This may be because the model can capture more higher-order correlations as the layer number increases. However, it may overfit the training data if too many layers are used, leading to decreased performance.

4.4.2 ***Impact of Prefix Length.*** We experiment with various lengths of prefix sequences to investigate the impact of the prefix sequence length. The setups of prefix sequence lengths and the results are shown in Figure 4. We can observe that the model's performance increases with the prefix sequence length and reaches its best when the prefix sequence length is 50. Short prefix sequences may contain too much noise, as the receptive field can't cover the whole user sequence. For example, when the length is 1, the model only captures the behavior pattern based on the latest behavior. Short prefix sequences have many false clicks and short-term fluctuations in user behavior, which are inconsistent with actual short-term intentions, and the model cannot capture long-term patterns of user behavior, thereby affecting the model's performance. On the other hand, long prefix sequences with the loss of some fine-grained information lose the advantage over short-term patterns.

4.4.3 ***Impact of Prefix-Target Pair Count.*** We conduct experiments with different numbers of prefix sequences and target item

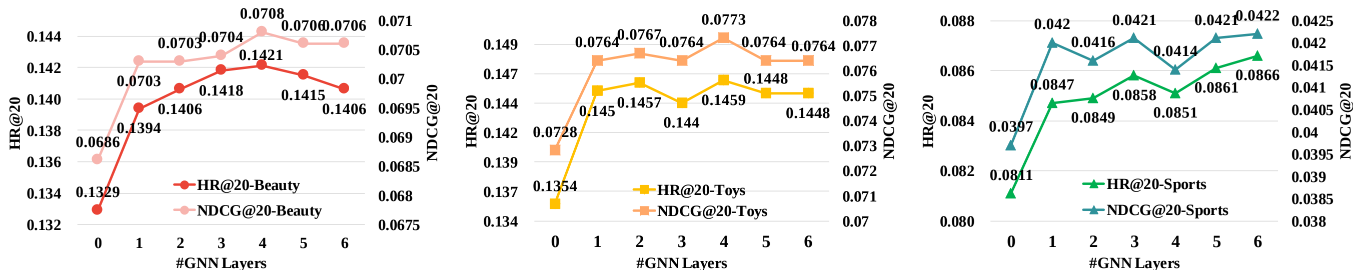

Figure 3: The impacts of different numbers of GNN Layers.

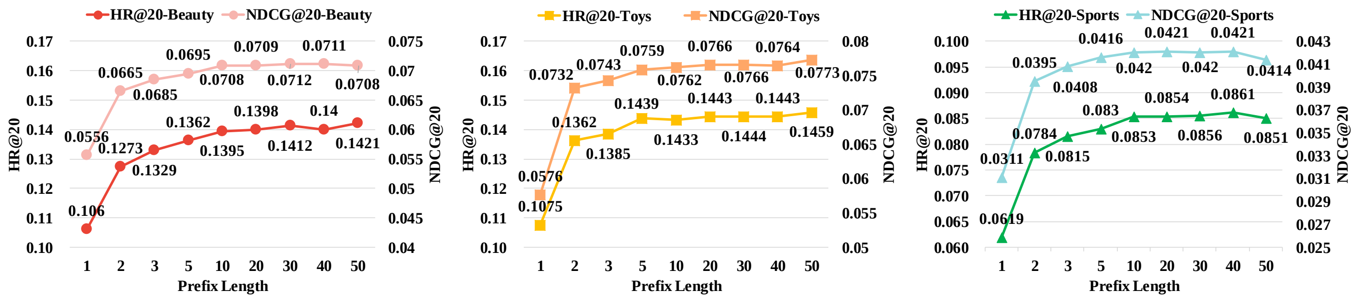

Figure 4: Impact of using different subsequence lengths in sequential recommendation training and PTG construction.

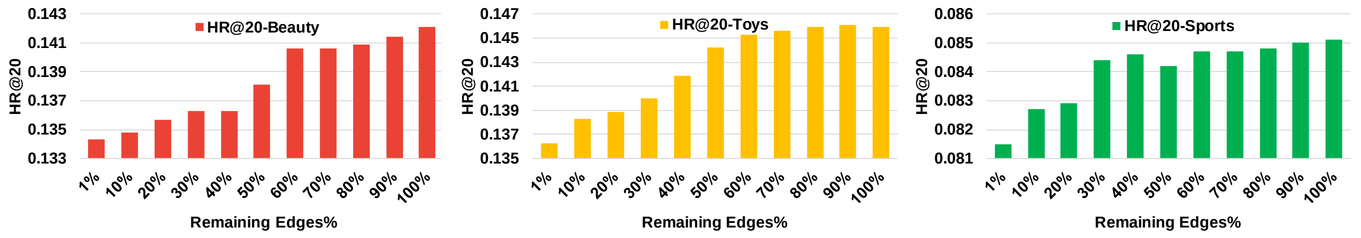

Figure 5: Impact of the number of observed prefixes in PTG. The x-axis represents the proportion of unmasked edges in PTG.

connections in the PTG. Specifically, random masks are applied to edges in the PTG to simulate that fewer sequences are observed. Results in Figure 5 show that the performance increases with the number of prefix sequences. It indicates that the more prefix sequences we have, the more information we can leverage to learn the transition and correlation relationships between items and prefix sequences. The PTSR is capable of fully utilizing this information to better capture the user's intention transition. Meanwhile, the performance reaches a plateau when the number of prefix sequences is sufficient, indicating that the PTSR has a strong generalization ability and can achieve good performance with a moderate number of prefix sequences.

5 CONCLUSION

In this paper, we tried to address the limitations of focusing on item-level transitions without fully exploring the user's intention transitions encapsulated within prefix sequences. To this end, we introduced a novel model called PTSR. It's designed to capture the transition and correlation relationships between targets and prefixes. PTSR segments user histories into multiple prefix sequences, each paired with a target item, thereby creating short interest regions reflecting user behavior patterns. It uses a Prefix-Target graph connecting nodes of target items with their prefix sequences based on their natural transition relationships. By applying GNN message propagation to this graph, it aggregates information from the neighbors of each node, enriching the representations of both items and prefix sequences. Extensive experiments demonstrated the superiority of the PTSR, which highlights its effectiveness in capturing the complex dynamics of user intention transitions and high-order relations. Future work could explore further enhancements to the graph construction process, incorporating additional contextual information or user-specific factors to further improve recommendation accuracy. Additionally, integrating more sophisticated GNN architectures or sequential recommenders could be investigated to push the boundaries of recommendation system performance.

6 ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (62172226) and the 2021 Jiangsu Shuangchuang (Mass Innovation and Entrepreneurship) Talent Program (JSSCBS20210200).

REFERENCES

[1] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 378–387.

[2] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and Maarten de Rijke. 2020. Improving End-to-End Sequential Recommendations with Intent-aware Diversification. In *CIKM*. 175–184.

[3] Yongjun Chen, Zhiwei Liu, Jia Li, Julian J. McAuley, and Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation. In *WWW*. 2172–2182.

[4] Sung Min Cho, Eunhyeok Park, and Sungjoo Yoo. 2020. MEANTIME: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *Proceedings of the 14th ACM Conference on recommender systems*. 515–520.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[7] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[8] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[9] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 161–169.

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

[11] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[15] Hengchang Hu, Wei Guo, Yong Liu, and Min-Yen Kan. 2023. Adaptive multi-modalities fusion in sequential recommendation systems. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 843–853.

[16] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

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

[20] Xuewei Li, Aitong Sun, Mankun Zhao, Jian Yu, Kun Zhu, Di Jin, Mei Yu, and Ruiguo Yu. 2023. Multi-Intention Oriented Contrastive Learning for Sequential Recommendation. In *WSDM*. 411–419.

[21] Yang Li, Tong Chen, Peng-Fei Zhang, and Hongzhi Yin. 2021. Lightweight Self-Attentive Sequential Recommendation. In *CIKM*. 967–977.

[22] Sijia Liu, Jiahao Liu, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2023. Autoseqrec: Autoencoder for efficient sequential recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1493–1502.

[23] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S Yu, Julian McAuley, and Caiming Xiong. 2021. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479* (2021).

[24] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *KDD*. ACM, 483–491.

[25] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR*. 43–52.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[27] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: Bring order to the web. In *Proc. of the 7th International World Wide Web Conf.*

[28] Xiuyuan Qin, Huanhuan Yuan, Pengpeng Zhao, Guanfeng Liu, Fuzhen Zhuang, and Victor S Sheng. 2024. Intent Contrastive Learning with Cross Subsequences for Sequential Recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 548–556.

[29] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*. 813–823.

[30] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM computing surveys (CSUR)* 51, 4 (2018), 1–36.

[31] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[33] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*. 1441–1450.

[34] Qiaoyu Tan, Jianwei Zhang, Jiangchao Yao, Ninghao Liu, Jingren Zhou, Hongxia Yang, and Xia Hu. 2021. Sparse-Interest Network for Sequential Recommendation. In *WSDM*. 598–606.

[35] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *DLRS@RecSys*. 17–22.

[36] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. 565–573.

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

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[39] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[40] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 169–178.

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

[42] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.

[43] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive Learning for Sequential Recommendation. In *ICDE*. 1259–1273.

[44] Zhao Yan, Xinguang Xiang, and ZeChao Li. 2022. Item correlation modeling in interaction sequence for graph convolutional session recommendation. *SCIENTIA SINICA Informationis* 52, 6 (2022), 1069–1082.

[45] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[46] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*. 1893–1902.