Next-item Recommendation with Sequential Hypergraphs

Jianling Wang
Texas A&M University
jlwang@tamu.edu

Kaize Ding
Arizona State University
kaize.ding@asu.edu

Liangjie Hong
LinkedIn Inc.
liahong@linkedin.com

Huan Liu
Arizona State University
huan.liu@asu.edu

James Caverlee
Texas A&M University
caverlee@tamu.edu

ABSTRACT

There is an increasing attention on next-item recommendation systems to infer the dynamic user preferences with sequential user interactions. While the semantics of an item can change over time and across users, the item correlations defined by user interactions in the short term can be distilled to capture such change, and help in uncovering the dynamic user preferences. Thus, we are motivated to develop a novel next-item recommendation framework empowered by sequential hypergraphs. Specifically, the framework: (i) adopts hypergraph to represent the short-term item correlations and applies multiple convolutional layers to capture multi-order connections in the hypergraph; (ii) models the connections between different time periods with a residual gating layer; and (iii) is equipped with a fusion layer to incorporate both the dynamic item embedding and short-term user intent to the representation of each interaction before feeding it into the self-attention layer for dynamic user modeling. Through experiments on datasets from the ecommerce sites Amazon and Etsy and the information sharing platform Goodreads, the proposed model can significantly outperform the state-of-the-art in predicting the next interesting item for each user.

ACM Reference Format:
Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item Recommendation with Sequential Hypergraphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401133

1 INTRODUCTION

In online platforms with millions of available items and churn in new items, recommendation systems act as an essential component to connect users with interesting items. From ecommerce platforms to streaming services to information sharing communities, recommenders aim to accurately infer the preferences of users based on their historical interactions, like purchases, views, and follows. In a promising direction, many recent efforts have shown

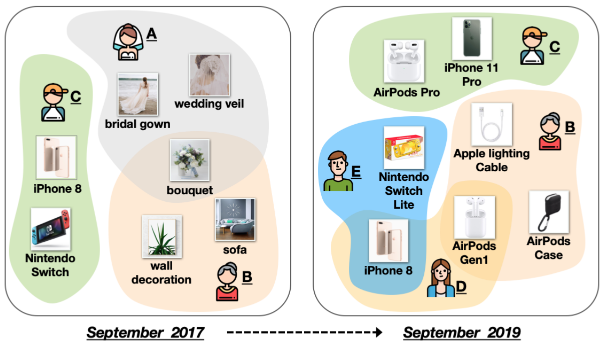

Figure 1: The meaning of an item at a certain time period can be revealed by the correlations defined by user interactions in the short term. And the meaning of an item can change over time and change across users. Such dynamics can help to uncover the preference patterns of users.

good success in next-item recommendation which aims to predict a user's next actions based on the sequential interactions in the past [22, 25, 30, 35, 37, 39, 44].

A critical issue is how items are treated in such models. Specifically, for a certain time period in next-item recommendation, we adopt the view that *the meaning of an item can be revealed by the correlations defined by user interactions in the short term.* As shown in Figure 1, the iPhone 8 was purchased together with several other up-to-date devices at the time it was released (like a Nintendo Switch) in 2017, indicating that it was a hot new technology item at that time. Once a new version is released in 2019 like the iPhone 11, the iPhone 8 becomes a budget choice since it may be purchased with other devices that are also budget-priced (e.g., the Lite version of the Nintendo Switch or early generation AirPods). In the same way, we can infer that the bouquet purchased by User *A* was for a wedding since she also purchased items typically associated with weddings. To capture these changes in item semantics, we propose to model such short-term item correlations in a *hypergraph* [1, 11], in which each *hyperedge* can connect multiple nodes on a single edge. In this regard, while each node in the hypergraph denotes an item, a hyperedge can connect the set of items a user interacts with in the short time period altogether.

However, it is non-trivial to extract expressive item semantics from the item-correlation hypergraph. On the one hand, the item correlations encoded by the hyperedges are no longer dyadic (pairwise), but rather triadic, tetradic or of a higher-order. Such complex

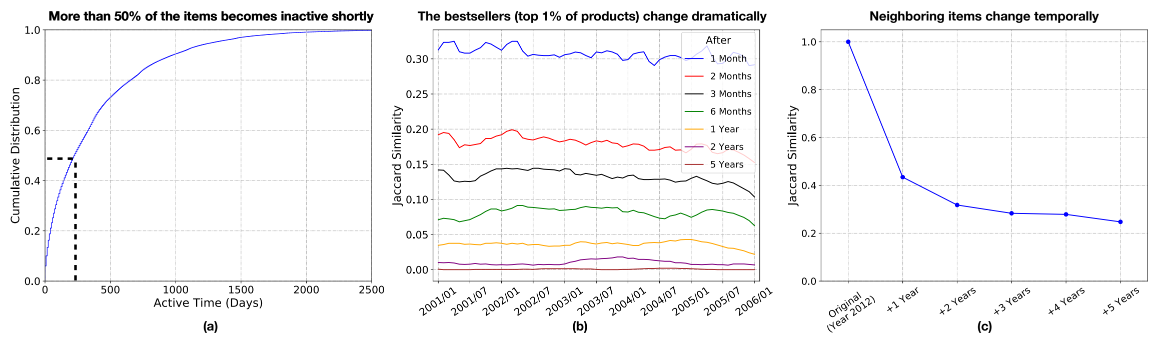

Figure 2: (a) The majority of listings (products) on Etsy become inactive in a year. (b) The overlap of monthly Bestsellers in Amazon decreases as the time gap grows larger (i.e., from 1 month to 5 years). (c) The neighboring books (books with large co-occurrence) on Goodreads are changing as time goes on.

relationships cannot be handled by conventional methods since they only focus on pairwise associations; on the other hand, the item semantics could propagate over multiple hops. For example, in Figure 1 (Sept 2019), though not purchased by the same user, the iPhone 8 is also related to the Apple Lightning cable with a 2-hop connection to it. Thus it necessitates a design to effectively exploit the hypergraph for learning expressive item semantics.

Furthermore, how to capture the dynamic meanings of items is another challenge for next-item recommendation, since the semantics of an item can change *over time* and *across users*. And such change can help to uncover the preference patterns of users. As illustrated in Figure 1, User *C* purchasing the iPhone 8 in 2017 gives evidence that User *C* chases the latest devices; whereas User *D* purchasing the iPhone 8 in 2019 indicates that User *D* is looking for a deal. Although the item is the same in both cases, the fundamental semantics of the iPhone 8 have changed. Even at a single timepoint, an item can carry different meanings to different users. For example, a bouquet of flowers for User *B* in Figure 1 can reflect home decoration, whereas the same bouquet for User *A* can reflect a wedding. Though there are previous works in next-item recommendation treating items as dynamic [7, 35, 39], they usually model the variation of an item as a function of time. How to capture the aforementioned two perspectives – change over time and change across users – is vital for high-quality next-item recommendations, but still remains to be explored.

To tackle the aforementioned challenges, we propose *HyperRec*, a novel end-to-end framework with sequential ***Hyper***graphs to enhance next-item ***Rec***ommendation. To untangle the short-term correlations at different time periods, HyperRec truncates the user interactions based on the timestamp to construct a series of hypergraphs. With a hypergraph convolutional network (HGCN), HyperRec is able to aggregate the correlated items with direct or high-order connections to generate the dynamic embedding at each time period. To model the influence of item embeddings in the past time periods, we develop a residual gating layer to combine the dynamic item embeddings of the previous time period with the static item embeddings to generate the input for the HGCN. With change happening both over time and across users, the resulting embeddings from the HGCN will be fed into a fusion layer to generate the final representation for each specific user-item interaction incorporating both the dynamic item embedding and short-term user intent. In personalized next-item recommendation, the dynamic user preferences can be inferred from the sequence of interactions from the user. Thus we use a self-attention layer to capture the dynamic user patterns from the interaction sequences. While predicting a user's preference on an item, both the static and the most recent dynamic item embedding are considered. We summarize our contributions as below:

- We investigate the dynamics of items from two perspectives – change over time and change across users – and uncover the importance in exploiting the short-term correlations between items for improving next-item recommendation.
- We are motivated to develop a novel next-item recommendation framework with sequential hypergraphs to generate dynamic item embeddings incorporating the short-term correlations between items. Two of the unique aspects of the framework are a residual gating layer to control the residual information from the past, and a fusion layer to encode each interaction with both the dynamic item embedding and short-term user intent for sequential pattern modeling.
- With extensive experiments on datasets covering different online platforms including ecommerce websites (Amazon and Etsy) and an information sharing community (Goodreads), the proposed model outperforms state-of-the-art models in providing Top-K next-item recommendation.

2 MOTIVATION

In this section, we conduct an initial investigation with data sampled from three online platforms – the ecommerce sites Amazon and Etsy and the information sharing platform Goodreads (see Section 4.1 for details of these three datasets). We explore the dynamic patterns of items and correlations between them from both the long-term and short-term perspectives.

Items emerge and disappear frequently. First, we examine the "lifecycle" of items in Etsy, which is one of the largest ecommerce platforms selling hand-crafted items. In Figure 2 (a), we summarize the active time, meaning the time gap between the first purchase and the last purchase, for all the items listed on Etsy from 2006 to 2018. We find that more than half of the products in Etsy become inactive (that is, they fall out of stock or are replaced by upgraded models) in less than one year. A similar pattern can be found in other online platforms. With the frequent emergence and disappearance of items, *short-term relationships may be critical for item modeling, whereas the relationships between items are unstable from a long-term perspective.*

The popularity of items changes rapidly along time. Second, we retrieve the Bestsellers (i.e., products ranked in the top 1% of purchases) on Amazon in each month from 2001 to 2005. We then calculate the Jaccard Similarity between the list of Bestsellers of each month with the Bestsellers after 1 month, 2 months, 3 months, 8 months, 1 year or more. In Figure 2 (b), as illustrated by the blue line, the intersection of Bestsellers between consecutive months is only around 30%. And there is little overlap between the list of Bestsellers after a gap of 6 months (with Jaccard similarity less than 10%). While the popularity of an item can reflect how the community views the item, *the change in the list of Bestsellers along time indicates that the meaning of items in the community can change along time.*

The co-occurrence of items changes temporally. Finally, we turn to the items in Goodreads, a platform in which users share their thoughts on books. Each user has a sequence of items that the user has interacted with via rating, tagging or commenting in chronological order. We split the sequences of items the users have interacted with based on the timestamps (by year) and train different item embedding models with sequences in different years. Following the idea in [6, 12, 35], we adopt word2vec [23] to generate embeddings of books based on the co-occurrence of items (i.e., books read by a user consequently). Based on these embeddings, we find the Top-10 neighbors of each book in different years. Then we calculate the Jaccard similarity between neighbors of each book in 2012 with its neighbors in 1 to 5 years later and show the average results in Figure 2 (c). We find that the similarity between neighbors in 2012 and 2013 for books is 40% and the similarity keeps decreasing as the time gap become larger. That is, *the relationships between items are changing along time and the variations are larger the longer the time gap.*

In summary, relationships between items are changing from the long-term perspective, leading to the change in the semantic meanings of items. Thus we are motivated to exploit the short-term correlations between items while modeling their dynamic patterns for next-item recommendation.

3 HYPERREC

In this section, we propose a novel end-to-end next-item recommendation framework empowered by sequential hypergraphs to incorporate the short-term item correlations while modeling the dynamics over time and across users. We will start with the problem setting of next-item recommendation. Then we introduce the details of the proposed HyperRec, centered around three guiding research questions: **RQ1** How to define correlations between items with a hypergraph structure and how to effectively incorporate the short-term item correlations into dynamic item embeddings by considering multi-hop connections between items? **RQ2** While the meaning of items in the past can hint on their characteristics in the future, how to link the embedding process at different time periods to connect how the residual information flows between consecutive time periods? **RQ3** How to fuse the short-term user intent with the dynamic item embedding to represent each interaction in a user interaction sequence for dynamic user preference modeling?

3.1 Problem Setting

We use $\mathbf{U} = \{u_1, u_2, ..., u_N\}$ to represent the set of N users and $\mathbf{I} = \{i_1, i_2, ..., i_P\}$ to represent the set of P items in a platform. We consider the set of Q different timestamps $\mathbf{T} = \{t_1, t_2, ..., t_Q\}$. Each timestamp $t_n \in \mathbf{T}$ is the equivalent of a certain short time period. For each user, we sort the list of items user u has interacted with in chronological order as $\mathbf{L}^u = \{(i_1^u, t_1^u), (i_2^u, t_2^u), ..., (i_{|\mathbf{L}^u|}^u, t_{|\mathbf{L}^u|}^u)\}$, in which (i_n^u, t_n^u) denotes that u interacted with item i_n^u at t_n^u, $t_n^u \in \mathbf{T}$. Items start with a set of static latent embeddings $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2 ..., \mathbf{e}_P]$, each of which is a trainable embedding associated with the item ID but unchanged for different users at different timestamps.

The goal of next-item recommendation is to predict the item that u will be interested in after $\mathbf{L}^u$. Note that to avoid data leakage, we use all the historical interactions on or before a cutting timestamp for model training. We aim to predict the next item each user will interact with after the cutting timestamp.

3.2 Sequential Hypergraphs

Since the items purchased by a user in a short time period are correlated, it is vital to define appropriate connections among them. While users may interact with various numbers of items, the conventional graph structure usually only supports pairwise relations between items and is not fit for this case. Thus, we propose to model such short-term correlations with a hypergraph [1, 11], in which multiple items can be connected with one *hyperedge*. For the example in Figure 1, the hypergraph for Sept 2017 consists of 7 nodes (items) with 3 hyperedges. The three items purchased by User A are linked together by one hyperedge. Furthermore, besides the direct connections in the hypergraph, the high-order connections between items can also hint on their correlations. For example, in Figure 1 (Sept 2019), though not purchased by the same user, the iPhone 8 is also related to the Apple Lightning cable with a 2-hop connection. With a hypergraph convolutional network (HGCN), we can exploit both the direct and high-order connections to extract the short-term correlations between items. Meanwhile, an item should not be treated as discrete at different time periods, since its features in the past can hint on its features in the future. For example, although the iPhone 8 has fundamentally changed in meaning from 2017 to 2019 in Figure 1, the representation in 2019 should inherit some of the characteristics of the iPhone's representation in 2017. In the following, with hypergraph as a principled topology structure, we will discuss about how to effectively generate such dynamic item representations considering both the item correlations in the short term and the connections among different time periods.

Short-term Hypergraphs. To capture the item correlations for different time periods, we can split the user-item interactions into multiple subsets based on the timestamps. Let $\mathbf{G} = \{\mathbf{G}^{t_1}, \mathbf{G}^{t_2}, ..., \mathbf{G}^{t_Q}\}$ represent a series of hypergraphs. $\mathbf{G}^{t_n} = (\mathcal{V}^{t_n}, \mathcal{E}^{t_n}, \mathbf{W}^{t_n}, \mathbf{H}^{t_n})$ is constructed based on all the user-item interactions happening during time period t_n. $\mathcal{V}^{t_n} \subset \mathbf{I}$ represents the set of nodes in $\mathbf{G}^{t_n}$, that is all the items with interactions in t_n. And $\mathcal{E}^{t_n} \subset \mathbf{U}$ denotes the set of hyperedges, which is similar as all the users who have

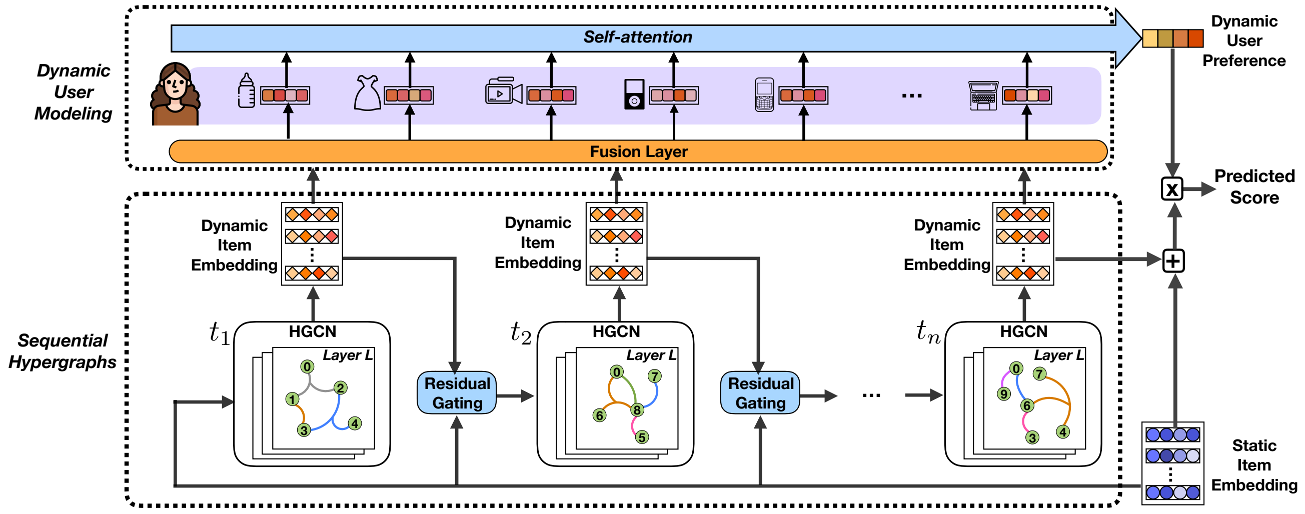

Figure 3: The structure of HyperRec: a series of hypergraphs are constructed based on item correlations at different time periods and the HGCN is able to capture the correlations in multi-hop connections. The resulting dynamic item embedding from the previous time period can influence the item embedding in the future via the Residual Gating layer. Both the dynamic item embedding and short-term user intent are fused to represent each interaction for dynamic user modeling.

interactions during t_n. Each $\mathbf{G}^{t_n} \in \mathbf{G}$ is associated with an incidence matrix $\mathbf{H}^{t_n}$ of size $|\mathcal{V}^{t_n}| \times |\mathcal{E}^{t_n}|$. It is also associated with a matrix $\mathbf{W}^{t_n}$, which is a diagonal matrix with $W^{t_n}_{\epsilon\epsilon}$ representing the weight of the hyperedge ϵ. In this work, we let all the hyperedges share the same weights and let $W^{t_n}_{\epsilon\epsilon} = 1, \forall \epsilon \in \mathcal{E}^{t_n}$. When $v \in \mathcal{V}^{t_n}$ is incident with edge ϵ during time period t_n (i.e., user ϵ purchased v at t_n), we have $H^{t_n}_{v\epsilon} = 1$, otherwise $H^{t_n}_{v\epsilon} = 0$. $\mathbf{D}^{t_n}$ and $\mathbf{B}^{t_n}$ are the diagonal degree matrices for vertex and hyperedge correspondingly, in which:

$$\mathbf{D}^{t_n}_{vv} = \sum_{\epsilon=1}^{|\mathcal{E}^{t_n}|} W^{t_n}_{\epsilon\epsilon} H^{t_n}_{v\epsilon} \quad \mathbf{B}^{t_n}_{\epsilon\epsilon} = \sum_{i=1}^{|\mathcal{V}^{t_n}|} H^{t_n}_{i\epsilon}$$

At different time periods, there will be a different set of user-item interactions, leading to hypergraphs with changing topology. We aim to extract the item semantics from each of the short-term hypergraphs by capturing item correlations.

Hypergraph Convolution Network (HGCN). At each time period, we aim to exploit the correlations among items for their temporally dynamic embeddings, in which the correlated items should be close with each other for the short time period. To achieve that, an item should aggregate information (i.e., latent representations) from all its neighboring items (i.e., items with connection to it). This naturally fits the assumption of the convolution operation [2, 3, 11, 19] that more propagation should be done between connected items. Given that nodes in $\mathcal{V}^{t_n}$ have a set of initial latent representation $\mathbf{X}^{t_n,0} = [\mathbf{x}^{t_n,0}_1, \mathbf{x}^{t_n,0}_2, ..., \mathbf{x}^{t_n,0}_{|\mathcal{V}^{t_n}|}]$, the convolution operation can be defined as:

$$\mathbf{x}^{t_n,1}_i = \tau\Big(\sum_{v=1}^{|\mathcal{V}^{t_n}|} \sum_{\epsilon=1}^{|\mathcal{E}^{t_n}|} H^{t_n}_{i\epsilon} H^{t_n}_{v\epsilon} W^{t_n}_{\epsilon\epsilon} \mathbf{x}^{t_n,0}_v \mathbf{P}^0\Big)$$

in which $\tau(\cdot)$ represents the activation function (ReLu in our experiment). $\mathbf{P}^0$ represents the trainable weight matrix between the initial and the 1^{th} layer. This convolution operation will encode each hyperedge with all the nodes connected to it and then output the embedding for each node by aggregating information of all the hyperedges it is on. We can formulate this convolution process into a matrix form as:

$$\mathbf{X}^{t_n,1} = \tau(\mathbf{H}^{t_n}\mathbf{W}^{t_n}{\mathbf{H}^{t_n}}^T\mathbf{X}^{t_n,0}\mathbf{P}^0)$$

To prevent numerical instabilities caused by stacking multiple convolutional layers, we need to add in symmetric normalization. Then we end up with:

$$\begin{aligned}\mathbf{X}^{t_n,1} &= f(\mathbf{X}^{t_n,0}, \mathbf{H}^{t_n}, \mathbf{W}^{t_n}|\mathbf{P}^0)\\ &= \tau({\mathbf{D}^{t_n}}^{-1/2}\mathbf{H}^{t_n}\mathbf{W}^{t_n}{\mathbf{B}^{t_n}}^{-1}{\mathbf{H}^{t_n}}^T{\mathbf{D}^{t_n}}^{-1/2}\mathbf{X}^{t_n,0}\mathbf{P}^0)\end{aligned} \tag{1}$$

Here $f(\cdot)$ is used to denote the operation for one hypergraph convolutional layer to update each node with its one-hop neighbors. We can stack multiple convolution layers to recursively aggregate the information from high-order neighbors in the hypergraph. In such a hypergraph convolutional network (HGCN), The output from the L^{th} layer can be calculated as:

$$\mathbf{X}^{t_n,L} = f(\mathbf{X}^{t_n,(L-1)}, \mathbf{H}^{t_n}, \mathbf{W}^{t_n}|\mathbf{P}^{(L-1)})$$

The resulting $\mathbf{X}^{t_n,L}$ from layer L can inherit embeddings from previous layers to capture the propagation of item correlations in the hypergraph. While at different time periods, the topology of hypergraphs is changing, leading to dynamic item embeddings reflecting the short-term correlations at different time periods.

Residual Gating. While items are changing, there is still linkage between their features at different timestamps. Some characteristics of an item will retain from the last time period to the next time period. For example, items may have some intrinsic features that change smoothly or are unchanged at all times. In order to propagate the residual information from the previous time periods to the future, we introduce a residual gating to generate the initial embedding of each node by combining the set of dynamic embeddings for

$t_1,..., t_{n-1}$ with the static embedding. The initial embedding of item i at t_n can be calculated as:

$$\mathbf{x}_i^{t_n,0} = g\mathbf{x}_i^{t_{<n},L} + (1-g)\mathbf{e}_i, \quad g = \frac{e^{\mathbf{z}_R^T\sigma(\mathbf{W}_R\mathbf{x}_i^{t_{<n},L})}}{e^{\mathbf{z}_R^T\sigma(\mathbf{W}_R\mathbf{x}_i^{t_{<n},L})} + e^{\mathbf{z}_R^T\sigma(\mathbf{W}_R\mathbf{e}_i)}}$$

in which $\mathbf{W}_R$ and $\mathbf{z}_R$ is the transformation matrix and vector for the gate. $\sigma(\cdot)$ is the *tanh* function. We use $\mathbf{x}_i^{t_{<n},L}$ to denote the dynamic embedding from the most recent hypergraph before t_n for item i. If item i doesn't appear in any previous hypergraph, we ignore the residual component and let $\mathbf{x}_i^{t_n,0} = \mathbf{e}_i$. The value g calculated with the gating function is used to control the percentage of residual information that will be retained. With this residual gating, we connect the hypergraph sequentially, leading to the major component of HyperRec – the sequential hypergraphs (as in Figure 3). At each time period, each item will be initialized from both the static item embedding and residual information from the past. And then the HGCN can incorporate the short-term item correlations to generate the expressive dynamic item embedding.

3.3 Dynamic User Modeling

Short-term User Intent. As introduced in Figure 1, the short-term user intent can be inferred from all the items the user has interacted with in a certain time period. This naturally falls into the definition of the *hyperedge* which accounts for all the items a user has interacted with in the short-term altogether. Thus moving one step forward, we can aggregate the dynamic node embedding on each hyperedge to infer each user's short-term intent with the following operation

$$\mathbf{U}^{t_n} = \tau(\mathbf{B}^{t_n -1/2}\mathbf{H}^{t_n T}\mathbf{D}^{t_n -1/2}\mathbf{X}^{t_n,L}\mathbf{P}^L) \tag{2}$$

The resulting matrix $\mathbf{U}^{t_n} = [\mathbf{u}_1^{t_n}, \mathbf{u}_2^{t_n},..., \mathbf{u}_{|\mathcal{E}^{t_n}|}^{t_n}]$ can be regarded as an assembly of short-term user intents at t_n.

Fusion Layer. Then we want to incorporate both the dynamic item embedding and the short-term user intent for a more expressive representation of each interaction in the sequence. We propose the fusion layer as below to generate the representation of the interaction between user u and item i at t_n:

$$\begin{aligned} \mathbf{e}_{i,u}^{t_n} &= \alpha_u\mathbf{u}_u^{t_n} + \alpha_d\mathbf{x}_i^{t_n,L} + (1-\alpha_d-\alpha_u)\mathbf{e}_i \\ \alpha_u &= \frac{e^{\mathbf{z}^T\sigma(\mathbf{W}_F\mathbf{u}_u^{t_n})}}{e^{\mathbf{z}^T\sigma(\mathbf{W}_F\mathbf{u}_u^{t_n})} + e^{\mathbf{z}^T\sigma(\mathbf{W}_F\mathbf{x}_i^{t_n,L})} + e^{\mathbf{z}^T\sigma(\mathbf{W}_F\mathbf{e}_i)}} \\ \alpha_d &= \frac{e^{\mathbf{z}^T\sigma(\mathbf{W}_F\mathbf{x}_i^{t_n,L})}}{e^{\mathbf{z}^T\sigma(\mathbf{W}_F\mathbf{u}_u^{t_n})} + e^{\mathbf{z}^T\sigma(\mathbf{W}_F\mathbf{x}_i^{t_n,L})} + e^{\mathbf{z}^T\sigma(\mathbf{W}_F\mathbf{e}_i)}} \end{aligned} \tag{3}$$

in which $\mathbf{e}_i$ and $\mathbf{x}_i^{t_n,L}$ is the static and dynamic item embedding correspondingly, and $\mathbf{u}_u^{t_n}$ is the vector in the matrix generated by Equation 2 to indicate the short-term user intent at t_n. $\mathbf{W}_F$ and $\mathbf{z}$ is the transformation matrix and vector correspondingly. To avoid the overfitting problem, during training, for interactions happening at the same timestamp as what we want to predict, we feed in $\mathbf{u}_u^{t_{n-1}}$ and $\mathbf{x}_i^{t_{n-1},L}$ to the fusion layer while generating $\mathbf{e}_{i,u}^{t_n}$.

Self-attention. With the superior performance of self-attention layer (i.e., Transformer) in next-item recommendation compared with CNN, RNN and Markov Chains-based models (as shown in [18]), we adopt self-attention as the basic model to capture the dynamic pattern in interaction sequences. $\mathbf{e}_{i,u}^{t_n}$ can be treated as embedding for interaction between i and u at t_n.

Assume that we have a sequence of items user u has interacted with in chronological order $\mathbf{L}^u = ((i_1^u, t_1^u), (i_2^u, t_2^u), ..., (i_{|\mathbf{L}^u|}^u, t_{|\mathbf{L}^u|}^u))$. To represent the k^{th} interaction, we also take the position k into consideration. We use $\mathbf{o}_k^u = \mathbf{e}_{i_k^u,u}^{t_k^u} + \mathbf{p}_k$ to represent the interaction, in which $\mathbf{p}_k$ is the positional embedding of position k to characterize the order information.

Given embedding sequence $(\mathbf{o}_1^u, \mathbf{o}_2^u, ..., \mathbf{o}_{|\mathbf{L}^u|}^u)$, self-attention [33] is designed to generate the aggregation based on the similarities (attention scores) between the last element $\mathbf{o}_{|\mathbf{L}^u|}^u$ and each element in the sequence. Then the attention score between $\mathbf{o}_{|\mathbf{L}^u|}^u$ and $\mathbf{o}_j^u$ can be calculated as:

$$att(\mathbf{o}_{|\mathbf{L}^u|}^u, \mathbf{o}_j^u) = \frac{(\mathbf{W}_Q\mathbf{o}_{|\mathbf{L}^u|}^u)^T(\mathbf{W}_K\mathbf{o}_j^u)}{\sqrt{d}}$$

in which $\mathbf{W}_Q$ and $\mathbf{W}_K$ are transformation matrices and d is the dimension of the embedding. Then the attentive aggregation can be calculated as:

$$\mathbf{d}_u^{t_{|\mathbf{L}^u|}^u} = \sum_{j=1}^{|\mathbf{L}^u|} att(\mathbf{o}_{|\mathbf{L}^u|}^u, \mathbf{o}_j^u)(\mathbf{W}_V\mathbf{o}_j^u) \tag{4}$$

where $\mathbf{W}_V$ is a transformation matrix. Then the generated $\mathbf{d}_u^{t_{|\mathbf{L}^u|}^u}$ can represent the dynamic preference of user u after interacting with the sequence of items in $|\mathbf{L}^u|$ at $t_{|\mathbf{L}^u|}^u$.

3.4 Preference Prediction

While predicting the preference of users for items, we should take both the dynamic item embedding and the static item embedding into consideration:

$$\hat{y}_{u,i}^{t_{n+1}} = \mathbf{d}_u^{t_{<n+1}T}(\mathbf{x}_i^{t_{<n+1},L} + \mathbf{e}_i) \tag{5}$$

in which $\mathbf{d}_u^{t_{<n+1}}$ and $\mathbf{x}_i^{t_{<n+1},L}$ denotes the most recent dynamic user preference and dynamic item embedding generated before t_{n+1}. To train the model, we adopt Bayesian Pairwise Loss [27], in which we assume that a user prefers item that she has interacted with to items she hasn't interacted with. The loss is calculated as

$$L = \sum_{(u,t,i,j)\in\mathbf{C}} -\ln\delta(\hat{y}_{u,i}^t - \hat{y}_{u,j}^t) + \lambda||\theta||^2$$

in which $||\theta||^2$ denotes the L2 regularization and λ is used to control its weight. δ is the Sigmoid function. Each element (u, t, i, j) in the training set $\mathbf{C}$ is constructed with a ground truth tuple (u, t, i) (i.e., u interacted with i at time period t) with an item j that u did not interact with (a negative sample).

4 EXPERIMENTS

In this section, we conduct experiments to evaluate the performance of the proposed HyperRec over datasets sampled from three online platforms (Goodreads, Amazon and Etsy). Besides its overall performance in next-item recommendation, we further investigate the design of HyperRec via ablation tests and parameter analysis. In addition, we also examine whether HyperRec can capture both

Dataset	# Users	# Items	# Interactions	Density	Cutting Timestamp
Amazon	74,823	64,602	1,475,092	0.0305%	Jan 1, 18
Etsy	15,357	56,969	489,189	0.0559%	Jan 1, 18
Goodreads	16,884	20,828	1730,711	0.4922%	Jan 1, 17

Table 1: Statistics of the datasets.

the long-term and short-term patterns in the platforms based on its recommendation to users with various lifespans.

4.1 Data

In the experiments, we formulate the next-item recommendation problem under leave-one-out setting as in previous works [18, 31] and split the train-test data following the real-world scenario as in [35, 39]. Note that models are trained with only the interactions on or before a cutting timestamp. We use the first interaction of each user after the cutting timestamp for validation and the second interaction for testing. To explore the generalization of the proposed model, we sample data from three different online platforms. Summary statistics of these datasets are in Table 1.

Amazon. This is the updated version of a public Amazon dataset [24] covering reviews from Amazon ranging from May 1996 to October 2018. In order to explore the short-term item correlations among a set of diverse products, we mix the purchase data from different categories instead of conducting experiments per-category. We use the review timestamp to approximate the timestamp of purchasing. We remove items with fewer than 50 purchases. We keep users who purchased at least 5 items before the cutting timestamp and purchased at least 2 items after the cutting timestamp.

Etsy. The Etsy dataset contains purchase records from November 2006 to December 2018 for one of the largest ecommerce sites selling handmade items. For data preparation, we remove products with fewer than 50 transactions, and then filter out users with fewer than 5 transactions before 2018 or fewer than 2 transactions in 2018.

Goodreads. This Goodreads dataset [35] is from a book reading community in which users can tag, rate, and write reviews on books. We treat different types of interactions equally as implicit feedback on items. We keep users who interacted with more than 5 books before 2017 and at least 2 books in 2017. This dataset is denser than both Amazon and Etsy since the items (i.e., books) in such an information sharing platform are more stable and less likely to be replaced by new items as in ecommerce platforms (e.g., products can be replaced by upgraded models).

4.2 Experimental Settings

4.2.1 *Evaluation Metrics.*

Following the leave-one-out setting, in the test data, each user only relates to one item that the user interacts with after the cutting time. We adopt the commonly used metrics for next-item recommendation including Hit Rate (HIT@K), Normalized Discounted Cumulative Gain (NDCG@K) and Mean Reciprocal Rank (MRR) to evaluate the performance of each model for Top-K recommendation. As in previous work for Top-K recommendation [15, 18], we randomly select 100 negative items for each user and rank the item in the test set (positive item) with the negative items. The ranking is based on the predicted preference scores of each user generated by the recommendation system.

Since there is only one item in the test set for each user, hit rate is equal to recall, indicating whether the tested item appears in the Top-K list. The Ideal Discounted Cumulative Gain is equal to a constant for all users and thus can be ignored while calculating NDCG@K. Given that the tested item of user u is ranked r_u based on the predicted scores, $NDCG_u@K = \frac{1}{\log_2(1+r_u)}$ if $r_u \leq K$ and $NDCG_u@K = 0$ otherwise. Meanwhile, $HIT_u@K = 1$ if $r_u \leq K$ and $HIT_u@K = 0$ otherwise. According to the calculation, $NDCG@1$ is equal to $HIT@1$ in the leave-one-out setting. We report the average NDCG and Hit Rate across all the users in each platform. MRR measures the average rankings of the tested items and $MRR = \frac{1}{|U|}\sum_{u\in U}\frac{1}{r_u}$, in which U is the set of all the users for testing. In the following, we report the results for $K = 1$ and $K = 5$.

4.2.2 *Baselines.*

- **PopRec:** *Popularity Recommendation.* This simple method ranks items based on their popularity and recommends the top items.
- **TransRec**: *Translation-based recommendation* [14]. TransRec models the transitions between different items in the interaction sequences with user-specific translation operations.
- **GRU4Rec+**: *Recurrent Neural Networks with Top-k Gains* [16]. As an improved version of GRU4Rec [17], this model adopts a GRU to model sequential user behaviors with a new class of loss functions designed for improving Top-K gains.
- **TCN**: *A Simple Convolutional Generative Network for Next Item Recommendation* [44]. This baseline improves the typical CNN-based next-item recommendation models with masked filters and stacked 1D dilated convolutional layers for modeling long-range dependencies.
- **HPMN**: *Lifelong Sequential Modeling with Personalized Memorization* [25]. HPMN is powered by a hierarchical periodic memory network to capture multi-scale sequential patterns of users simultaneously, and thus can combine recent user behaviors with long-term patterns.
- **HGN**: *Hierarchical Gating Networks for Sequential Recommendation* [22]. This method contains a feature gating and an instance gating to hierarchically select the features and instance of items for user modeling while making next-item recommendation.
- **SASRec**: *Self-attentive Sequential Recommendation* [18]. It adopts the self-attention layer to capture the dynamic patterns in user interaction sequences. It can be treated as a simplified version of the dynamic user modeling component in HyperRec to use the static item embeddings to represent each interaction.
- **BERT4Rec**: *Sequential Recommendation with Bidirectional Encoder Representations from Transformer* [31]. This baseline utilizes a bi-directional self-attention module to capture the context information in user historical behavior sequences from both left and right sides.

Besides the baselines above, we also compare the proposed model with its variants in Section 4.4 as our ablation test cases.

4.2.3 *Parameters.*

Our experiments are conducted on a server machine equipped with a 12 GB Nvidia TITAN Xp GPU. We set the maximum sequence length to be 50 for all the datasets. For fair comparison, the negative sampling rate is set to be 1 for all the

Metrics	Datasets	**PopRec**	**TransRec**	**HPMN**	**TCN**	**GRU4Rec+**	**BERT4Rec**	**HGN**	**SASRec**	**HyperRec**	Improv.
NDCG@1/ HIT@1	Amazon	0.0423	0.0533	0.0771	0.0783	0.0983	0.1011	0.1012	0.1051	0.1215∗	20.03%
	Etsy	0.0677	0.4201	0.3746	0.3816	0.3916	0.4338	0.4379	0.4477	0.4725∗	7.90%
	Goodreads	0.0776	0.2174	0.2229	0.2069	0.2360	0.2366	0.2447	0.2643	0.2878*	17.62%
NDCG@5	Amazon	0.1026	0.1202	0.1663	0.1648	0.1989	0.2010	0.1981	0.2041	0.2264∗	12.60%
	Etsy	0.1386	0.5495	0.5096	0.5120	0.5307	0.5553	0.5698	0.5713	0.5946∗	4.37%
	Goodreads	0.1694	0.3752	0.3847	0.3593	0.4035	0.4073	0.4163	0.4326	0.4624∗	11.07%
HIT@5	Amazon	0.1633	0.1867	0.2543	0.2499	0.2963	0.2972	0.2918	0.3001	0.3272∗	10.08%
	Etsy	0.2084	0.6678	0.6300	0.6310	0.6566	0.6650	0.6885	0.6816	0.7047∗	2.35%
	Goodreads	0.2587	0.5234	0.5358	0.5009	0.5581	0.5643	0.5747	0.5865	0.6206∗	7.98%
MRR	Amazon	0.1204	0.1357	0.1780	0.1777	0.2073	0.2094	0.2070	0.2120	0.2328∗	11.19%
	Etsy	0.1526	0.5328	0.4920	0.4974	0.5131	0.5411	0.5519	0.5555	0.5780∗	4.73%
	Goodreads	0.1801	0.3624	0.3707	0.3495	0.3867	0.3896	0.3979	0.4146	0.4418∗	11.02%

Table 2: Comparison of Different Models. $*$ indicates that the improvement of the best result is statistically significant compared with the next-best result with $p < 0.01$.

models in the training process. That is, we couple each ground-truth tuple with a randomly sampled negative item.

For HPMN, GRU4Rec+, HGN and BERT4Rec, we use the implementations and settings as provided in the original papers. As for TCN and SASRec, we use the implementation provided in [30]. To achieve the best performance for each model, we grid search for the dropout rates in {0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, the regularization weight λ in {10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}}, the learning rate in {10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}} and the embedding size in {25, 50, 100, 150, 200}. For the model-specific hyper-parameters, we fine-tune them based on results in the validation set.

We implement HyperRec and all its variants in TensorFlow and adopt Adam as the optimizer. In the experiment, after the grid-search, the learning rate is set to be 0.001 and the batch size is set to be 5120. We set the embedding size to be 100 for all the datasets. We fine-tune the number of convolution layers in {1, 2, 3, 4, 5} and the granularity of time periods in {1, 3, 6, 12, 18} months for each dataset while reporting the results in Table 2.

4.3 Evaluation

We compare HyperRec with the baselines and the results are reported in Table 2. Under all the evaluation metrics, HyperRec can significantly outperform all the baselines in each of the datasets, which demonstrates its effectiveness in improving next-item recommendation in realistic settings where items evolve over time.

As a pioneer for personalized next-item recommendation, TransRec can provide promising improvement compared with simply recommending the most popular items. However, TransRec treats users as a linear translation between consecutive items they purchase, which limits the model in dealing with the realistic problems that both users and items are changing. With the development of neural networks in capturing dynamic patterns in sequential data, there are lots of recent efforts in adopting these neural structure for next-item recommendation. HPMN consists of hierarchical memory networks to create lifelong profiles for users, in which each layer of the memory network is designed to capture periodic user preferences with a specific period. We find that HPMN outperforms TransRec by more than 30% in Amazon but appear to be weak on Etsy and Goodreads. Building on top of 1D dilated convolution layers, TCN shows its strength in modeling the short-term behaviors and outperforms HPMN in ecommerce for Etsy. It does not seem to be a good fit for scenario like Goodreads in which the long-term preferences are significant. As an advanced version of GRU4Rec targeting Top-K recommendation, in Amazon and Goodreads, GRU4Rec+ can improve TCN and HPMN by conducting dynamic user modeling with GRU and adopting a loss function tailored to RNN-based models for Top-K recommendation. The newly proposed HGN is equipped with a novel feature-gating and an instance gating to enhance the short-term user modeling, and thus can outperform the aforementioned baselines. Both SASRec and Bert4Rec employ a self-attention layer to model the sequential user patterns. In BERT4Rec, by randomly masking items in the user sequences, it is able to train a bidirectional model for recommendation. However, it does not bring in huge improvement as in the original BERT applications for natural language processing since the right-to-left patterns in sequences are not necessarily informative for predicting dynamic user preferences.

Compared with the state-of-the-art, HyperRec can achieve its largest improvement in Amazon than in other datasets. The reason might be that HyperRec is able to fully extract the item characteristics from extremely sparse user-item interactions with the Hypergraph topology. Meanwhile, the outstanding performance of HyperRec in both ecommerce and information sharing platforms demonstrates that it can be generalized to various online scenarios.

4.4 Ablation and Parameter Study

In this section, we first conduct a series of ablation tests by removing or replacing the essential components in HyperRec to evaluate their effectiveness. Then we explore the performance of HyperRec with various number of convolution layers and different granularity of time periods to further investigate how it works in exploiting the short-term item correlations for next-item recommendation.

Ablation Test. We report the results of our ablation tests in Table 3. For fair comparison, results are all achieved with granularity of time periods to be 12-months and HGCN containing 2 layers when there are hypergraphs in the models. First of all, HyperRec can achieve the best performance compared to any of its variants for all the datasets, indicating the effectiveness of its design.

To evaluate the effectiveness of the hypergraph structure, in (3), we assign each user and each item with different latent embeddings

Architecture	Amazon	Etsy	Goodreads
(1) *HyperRec*	0.1215	0.4712	0.2809
(2) *Static Item Embedding*	0.1051	0.4477	0.2643
(3) *Replace Hypergraph*	0.0978	0.4588	0.2576
(4) *(-) Residual*	0.1169	0.4591	0.2626
(5) *(-) Dynamic Item Embedding*	0.1131	0.4646	0.2789
(6) *(-) Short-term User Intent*	0.1147	0.4616	0.2709
(7) *(-) Dynamic in Prediction*	0.1151	0.4703	0.2746

Table 3: Results for Ablation Test under HIT@1/NDCG@1. (-) denotes removing the specific component.

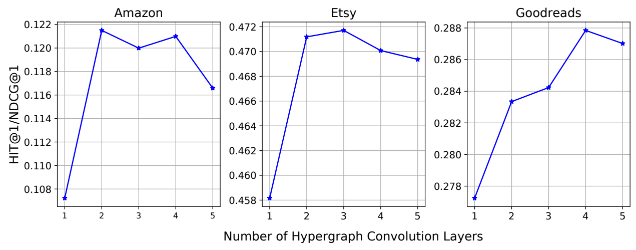

Figure 4: Performance comparison with different number of HGCN layers under HIT@1/NDCG@1.

at different time periods as the dynamic item embeddings and short-term user intents. That is, instead of exploiting the short-term item correlations with a hypergraph as in HyperRec, we use these time-dependent embeddings to encode the change in the platforms. We find that there is a huge drop in performance. In Amazon and Goodreads, the performance of (3) is even worse than that of (2) which uses static item embeddings. One reason is that the user-item interactions at each time period are too sparse to sufficiently train the time-dependent embedding directly. But hypergraph with HGCN is able to fully extract the effective correlations between items from the multi-hop connections at each time period.

Then we turn to each of the components empowered by the dynamic item embedding in HyperRec to examine its contributions to next-item recommendation. In (4), by removing the residual component, the initial embedding of the hypergraph only consists of the static item embedding. While the performance drops in all the datasets, we find it brings in the largest loss of performance in Goodreads. Therefore it is important to connect the dynamic item embedding at different time periods via controlling the residual information from the past. For (5) and (6), we remove $\mathbf{x}_i^{t_n,L}$ and $\mathbf{u}_u^{t_n}$ in the fusion layer respectively to examine how the resulting dynamic item embedding and short-term user intent contribute to the meaning of a specific interaction in dynamic user preference modeling. Since (5) and (6) achieve a similar performance in all of the datasets, we may conclude that both components are important for capturing the meaning of an interaction. In (7), we remove the dynamic item embedding (from the most recent time period) while calculating the preference score with Equation 5. We find that this component contributes a lot for Amazon and Goodreads. However, on Etsy, items are more sensitive to instant gifting events (e.g., Christmas, anniversaries), so the dynamic item embedding from the last time period may not be able to provide great insight into the current time period.

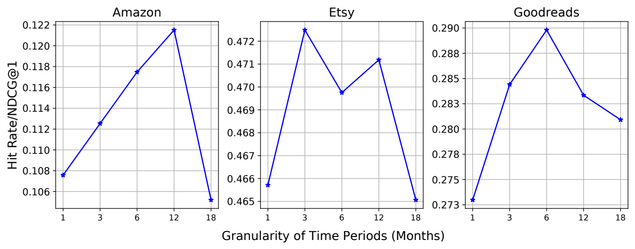

Figure 5: Performance comparison with various time granularity under HIT@1/NDCG@1.

4.5 Different User Groups

Number of HGCN Layers. To explore how the high-order connections in the hypergraph can help to uncover hidden item correlations and thus contribute to the final recommendation for different online platforms, we compare the performance of HyperRec by varying the number of hypergraph convolutional layers (in Figure 4). When there is one convolution layer for the sequential hypergraphs, each dynamic item embedding aggregates only information from items connected with them directly by the hyperedge. At this stage, HyperRec can outperform its variants considering only static item embedding, which illustrates the necessity of exploring the short-term item correlations and adopting dynamic embedding in next-item recommendation. Furthermore, by stacking two HGCN layers, it can bring in significant improvement compared with a model with just one convolution layer. We can infer that hypergraph and HGCN are effective options for extracting expressive item semantics in the short term. And it is important to take the high-order neighboring information in hypergraph into consideration. However, for Etsy and Amazon, since the data is very sparse, it is not necessary to further increase the number of convolutional layers. 2-3 HGCN layers are enough for extracting the item semantics at different time periods. However, while Goodreads contains comparatively more interactions in each graph, more convolutional layers can further improve the embedding process. This demonstrates the effectiveness of hypergraph and HGCN in modeling the short-term item correlations for next-item recommendation.

Change of Time Granularity. An important parameter which can control how sensitive HyperRec is to the change over time is the granularity of the time period. Thus in Figure 5, we show the performance of the proposed model by varying the granularity from 1 month to 18 months. When the granularity is small, we find that the model cannot achieve the best performance since the interactions are extremely sparse and not sufficient for building up a set of expressive item embeddings. While enlarging the granularity, we find that the performance of HyperRec is increasing in all the datasets. In Amazon, it reaches the best performance when the granularity is set to be 12-months. However, for Etsy, the optimized granularity is smaller since the products sold on Etsy (i.e., handcrafted items) are in higher volatility than products on Amazon. In Goodreads, the optimized granularity is around 6-months, which is smaller than that for the other datasets since there are more interactions for each time period in Goodreads for the dynamic item embedding. If we further enlarge the granularity, the performance

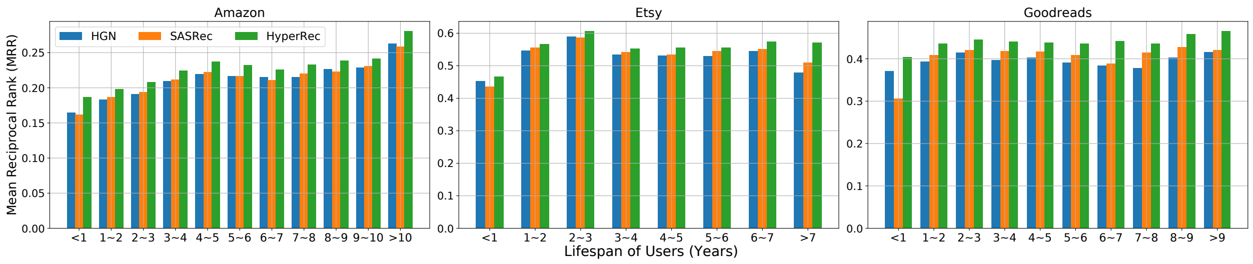

Figure 6: Performance comparison for users with different lifespans.

will decrease since it underestimates the change of items and may introduce noise to the model.

To further explore the performance of the proposed model in both long-term and short-term scenarios, we compare HyperRec with the top-2 baselines, HGN and SASRec, for users with various lifespans in the platforms (in Figure 6). Here, we calculate the time gap between the last interaction and the first interaction for each user as his/her lifespan in the platform. We find that HGN works better than SASRec for users with a short lifespan (less than one year), while SASRec can outperform HGN in modeling the users who are active for longer time in the platforms. However, we find that HyperRec significantly outperforms the baselines for users with both short and long lifespans. And it can achieve comparatively larger improvement for users with longer lifespans, indicating that HyperRec is superior in capturing the long-term patterns while taking the short-term correlations into consideration.

5 RELATED WORK

Next-item Recommendation. Next-item recommendation has been a promising research topic recently. Compared with recommendation systems treating users as static, it usually updates a user's status after each of her interactions and generates predictions relying on the relationships between items consumed sequentially. Some works focus on recommendation for short-term interaction sessions without user identification, which usually assume that items in a session are highly correlated with each other and center around an intense intent [26, 41].

Another line of research models user preferences with historic item sequences spanning a longer period of time. Pioneering works adopt Markov Chains [28] and translation-based [14] methods to model the transition between items that a user interacted with sequentially. Recently, there are lots of efforts in applying different neural networks to capture users' dynamic preferences from their sequential behaviors. GRU4Rec [17] utilizes a Gated Recurrent Neural Network (GRU) to investigate users' sequential interactions and then GRU4Rec+ [16] is proposed as a modified version with a new class of loss function designed for the Top-K recommendation. Meanwhile, Convolutional Neural Networks (CNN) are adopted by [32, 44] to capture the sequential patterns of users' historic interactions. While self-attention layer (transformer) [33] is proposed to be an effective replacement for RNN and CNN in handling sequential data, it is adopted in SASRec [18] to extract user preferences from the interactions in the past. However, these methods focus on modeling the sequential patterns without considering the temporal effects, leading to similar latent representations for interactions happening at different time periods or from various users.

There are previous efforts paying attention to the temporal effects in the design of recommendation systems. In TimeSVD++ [20], to preserve the temporal dynamics, they train various SVD models with ratings at different time periods. Considering both users and items are evolving along time, the works in [29, 35, 39] utilize parallel RNNs to model the sequential patterns of users and items separately and aim to predict how a user will rate the items at different timestamps. [7] proposes to generate coevolutionary feature embeddings for users and items with a model combining an RNN and point process. These methods are designed for explicit feedback sequences (i.e., ratings) and need to rely on precise timing information. Thus they are not suitable for handling scenarios with implicit feedback and sparse timestamps.

Neural Graph-based Recommendation. There is an increasing attention on exploiting graph structures for various recommendation scenarios with the recent advance in neural graph embedding algorithms [8, 9, 13, 19, 34]. Many of these works make use of the high-order connections in a static graph to generate enriched latent representations for users or items. In social recommendation, social connections between users can be investigated with GNN to model the propagation of user preference in social networks [10, 36, 40]. Differently, PinSage [19] proposes to generate item embeddings on a graph constructed with item-item connections, which can be applied for downstream recommendation. In addition, there are also works focusing on the user-item interaction graph [4, 38] in which they construct a static graph connecting users and items based on their interaction. However, these methods are not designed for capturing the sequential patterns in recommendation systems.

To model the temporally dynamic patterns and predict for future behaviors, Session-based Temporal Graph (STG) [42] is proposed to connect users, items and sessions in a graph. With random walk process starting from different type of nodes (user/session), it is able to model users' long-term and short-term preferences for recommendation. The work of [30] consists of an RNN to capture dynamic user behaviors and a graph attention layer to model the social influence on a static user-user graph. SR-GNN [41] proposes to construct various graphs of items with session sequences and use GNN to extract item co-occurrences from those session graphs. It generates next-click prediction based on attentive aggregation of item embedding in a session.

As a generalization of the ordinary graph in which each hyperedge can encode the correlations among various numbers of objects, hypergraph has been adopted to unify various types of contents for context-aware recommendation. In terms of modeling the correlations among various types of objects, there are early efforts [5, 21, 43, 45] in applying hypergraphs to assist conventional collaborative filtering for incorporating context information. In [5], in order to integrate both the social relationships and music contents for music recommendation, they propose to use hypergraph to model the relations among various types of objects (e.g., users, groups, music tracks, tags, albums) in music social communities. Similarly, the work of [21] models the correlations among readers, articles, entities and topics with a hypergraph for personalized news recommendation. These methods are designed based on the properties of the specific communities and can not be easily generalized to the task of next-item recommendation.

6 CONCLUSION

In this work, we explore the dynamic meaning of items in real-world scenarios and propose a novel next-item recommendation framework empowered by sequential hypergraphs to incorporate the short-term item correlations for dynamic item embedding. With the stacking of hypergraph convolution networks, a residual gating and the fusion layer, the proposed model is able to provide more accurate modeling of user preferences, leading to improved performance compared to the state-of-the-art in predicting user's next action for both ecommerce (Amazon and Etsy) and information sharing platform (Goodreads). In the future, we are interested in investigating how to transfer the dynamic patterns across platforms or across domains for an improved predictive performance.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant IIS-1841138.

REFERENCES

[1] Sameer Agarwal, Kristin Branson, and Serge Belongie. 2006. Higher order learning with graphs. In *ICML*.

[2] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *NeurIPS*.

[3] Song Bai, Feihu Zhang, and Philip HS Torr. 2019. Hypergraph Convolution and Hypergraph Attention. *arXiv preprint arXiv:1901.08150* (2019).

[4] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[5] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music recommendation by unified hypergraph: combining social media information and music content. In *MM*.

[6] Zhiyong Cheng, Jialie Shen, Lei Zhu, Mohan S Kankanhalli, and Liqiang Nie. 2017. Exploiting Music Play Sequence for Music Recommendation.. In *IJCAI*.

[7] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Recurrent coevolutionary latent feature processes for continuous-time recommendation. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*.

[8] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep anomaly detection on attributed networks. In *SDM*.

[9] Kaize Ding, Yichuan Li, Jundong Li, Chenghao Liu, and Huan Liu. 2019. Feature Interaction-aware Graph Neural Networks. *arXiv preprint arXiv:1908.07110* (2019).

[10] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *WWW*.

[11] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *AAAI*.

[12] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *KDD*.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[14] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *RecSys*.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[16] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*.

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

[18] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*.

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

[20] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*.

[21] Lei Li and Tao Li. 2013. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *WSDM*.

[22] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical Gating Networks for Sequential Recommendation. In *KDD*.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

[24] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *EMNLP-IJCNLP*.

[25] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *SIGIR*.

[26] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. In *AAAI*.

[27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[28] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*.

[29] Qingquan Song, Shiyu Chang, and Xia Hu. 2019. Coupled Variational Recurrent Collaborative Filtering. In *KDD*.

[30] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based Social Recommendation via Dynamic Graph Attention Networks. In *WSDM*.

[31] 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. *CIKM*.

[32] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[35] Jianling Wang and James Caverlee. 2019. Recurrent Recommendation with Local Coherence. In *WSDM*.

[36] Jianling Wang, Kaize Ding, Ziwei Zhu, Yin Zhang, and James Caverlee. 2020. Key Opinion Leaders in Recommendation Systems: Opinion Elicitation and Diffusion. In *WSDM*.

[37] Jianling Wang, Raphael Louca, Diane Hu, Caitlin Cellier, James Caverlee, and Liangjie Hong. 2020. Time to Shop for Valentine's Day: Shopping Occasions and Sequential Recommendation in E-commerce. In *WSDM*.

[38] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*.

[39] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*.

[40] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. *arXiv preprint arXiv:1904.10322* (2019).

[41] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*.

[42] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. 2010. Temporal recommendation on graphs via long-and short-term preference fusion. In *KDD*.

[43] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. 2019. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *WWW*.

[44] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *WSDM*.

[45] Yu Zhu, Ziyu Guan, Shulong Tan, Haifeng Liu, Deng Cai, and Xiaofei He. 2016. Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing* 216 (2016), 150–162.