Pairwise Interactive Graph Attention Network for Context-Aware Recommendation

Yahui Liu[1], **Furao Shen**[1], **Jian Zhao**[2]
[1] Department of Computer Science and Technology, Nanjing University, Nanjing, China
[2] School of Electronic Science and Engineering, Nanjing University, Nanjing, China
liuyahui@smail.nju.edu.cn, {frshen, jianzhao}@nju.edu.cn

Abstract

Context-aware recommender systems (CARS), which consider rich side information to improve recommendation performance, have caught more and more attention in both academia and industry. How to predict user preferences from diverse contextual features is the core of CARS. Several recent models pay attention to user behaviors and use specifically designed structures to extract adaptive user interests from history behaviors. However, few works take item history interactions into consideration, which leads to the insufficiency of item feature representation and item attraction extraction. From these observations, we model the user-item interaction as a dynamic interaction graph (DIG) and proposed a GNN-based model called Pairwise Interactive Graph Attention Network (PIGAT) to capture dynamic user interests and item attractions simultaneously. PIGAT introduces the attention mechanism to consider the importance of each interacted user/item to both the user and the item, which captures user interests, item attractions and their influence on the recommendation context. Moreover, confidence embeddings are applied to interactions to distinguish the confidence of interactions occurring at different times. Then more expressive user/item representations and adaptive interaction features are generated, which benefits the recommendation performance especially when involving long-tail items. We conduct experiments on three real-world datasets to demonstrate the effectiveness of PIGAT.

Introduction

Recommender systems (RS), aim to discover the preferred items for potential users, play an increasingly important role in practical applications such as E-commerce and social media. Typically, the core of recommendation systems is to predict user preference precisely, where the preference is usually reflected in rating, clicking, consuming and other user behaviors. When predicting, rich side information such as user profile, item profile and user behaviors is also available beyond the essential user ID and item ID. Context-aware recommendation systems (CARS) are designed to address these highly sparse categorical features to predict user preference more accurately, which has attracted widespread attention in both academia and industry (Rendle 2010; Cheng et al. 2016; Qu et al. 2016; Lian et al. 2018).

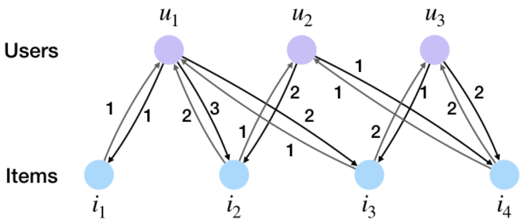


Figure 1: An illustration of the dynamic user-item interaction graph. Labels indicate the interaction order.

To obtain the goal of predicting user preference, it is critical to capture user interests and item attractions from numerous features and figure out whether they match each other. Recently, a series of prediction models pay specific attention to user interest representation by finely dealing with user history behaviors (Zhou et al. 2018b; Zhou et al. 2018a; Zhou et al. 2019; Feng et al. 2019). However, most CARS models pay less attention to extract item attractions from item historical interaction log. With the historical interaction log, people most likely to be attracted by the item could be captured, which benefits to measure the attraction of the item to a particular user and to enrich item features. These advantages greatly improve the recommendation performance especially for unpopular items since in this way the connections between unpopular items and their interacted users are established, so that more expressive item feature representations could be characterized. Moreover, both user interests and item attractions are dynamic, interactions occurring at different times have different confidence in representing user interests or item attractions. Thus, it should be taken into account to distinguish the confidence of different interactions.

Motivated by above observations and inspired by recent developments of graph neural networks (GNN) (Vaswani et al. 2017; Kipf and Welling 2017; Velickovic et al. 2018) which has the ability to generate hidden representations of graph nodes or the whole graph, we first model the dynamic user-item interactions as dynamic user-item interaction graph (DIG), and then propose Pairwise Interactive Graph Attention Network (PIGAT) to make full use of dynamic user-item interaction information and improve the prediction performance. As shown in Figure 1, DIG is a directed bipartite graph, each node represents a user or an item, and each directed edge represents an user-item interaction with a label indicates the interaction order in term of the head.

With DIG, our PIGAT uses the attention mechanism to capture two types of the importance of each interacted user or item: the importance to the head and the importance to the recommendation context. Considering these importance measurements together, PIGAT takes the weighted sum pooling to obtain the interactive head representation as well as adaptive interaction representation. Furthermore, PIGAT introduces confidence embeddings to distinguish the confidence of interactions occurring at different times. Hence in our architecture, important interactions have greater impacts on the hidden representations and the importance is measured under multiple criteria, which brings improvement of the representation ability of PIGAT.

The main contributions of this work are summarized as follows:

- We highlight the importance of the user-item interactions and propose a dynamic interaction graph to represent the dynamic interactions between users and items, which can also be taken as a generalization of user historical behaviors.
- We propose a GNN-based architecture that extracts expressive interactive representations from the interaction graph to improve the performance of content-aware recommendation especially for recommendation involving unpopular items.
- We employ confidence embedding to distinguish the interactions with different orders and design a novel initialization approach to make the embedding more trainable and effective.
- Our model achieves state-of-the-art performance in experiments on three real-world datasets and significantly outperforms other comparative models in context-aware recommendation task.

Preliminaries

In this section, we introduce the related concepts and works to our PIGAT, which includes context-aware recommendation, graph-based recommendation and long-tail item recommendation.

Context-Aware Recommendation

CARS models consider rich categorical side information besides basic user ID and item ID. Typically, each categorical feature is associated with an embedding to expressively represent the feature, and then complex feature combinations are learned from embeddings. Traditional models use fixed functions to model feature combinations. For example, FM (Rendle 2010) uses the inner-product to model the combination of each pair of features. Modern models often share the Embedding&MLP paradigm, in which feature combinations are learned through the MLP network. Such deep models capture complex feature combinations and bring significant improvements in the recommendation performance. Deep Crossing (Shan et al. 2016) concatenates all the embeddings and use residual units to learn feature combinations. Wide&Deep (Cheng et al. 2016) combines munually designed features and MLP generated features. DeepFM (Lian et al. 2018) combines low-order and high-order feature combinations.

Recently, attention mechanism (Bahdanau, Cho, and Bengio 2015; Vaswani et al. 2017) is introduced from Neural Machine Translation (NMT) field to learn the importance of different feature representations. AFM (Xiao et al. 2017) follows attention mechanism to distinguish the importance of different feature combinations. More models take effort on the user behavior representation, which improves the simply fixed-size representation used in YoutubeNet (Covington, Adams, and Sargin 2016). DIN (Zhou et al. 2018b) adaptively considers the relative importance of each user behavior to the candidate item. ATRank (Zhou et al. 2018a) uses the self-attention mechanism to model heterogeneous user behaviors.

GNN-Based Recommendation

In recommender systems, user-item interactions are often modeled as an undirected bipartite graph, where users and items are represented by two disjoint parts of the graph and each edge represents the interaction between its endpoints. Inspired by the recent progress in GNN, GC-MC (van den Berg, Kipf, and Welling 2017) applies the graph convolution network (GCN) (Kipf and Welling 2017) on user-item interaction graph to capture direct user-item relationship, NGCF (Wang et al. 2019) builds GNN-based embedding propagation layers to capture collaborative signal through the high-order connectivity. Such models benefit a lot from the strong node representation ability of GNN. However, they mainly focus on the mboxinner-graph feature representations and lack the ability to capture the relationship between dynamic interactions and recommendation context.

In this work, we modify the definition of the original interaction graph to satisfy the dynamic setting. As shown in Figure 1, the dynamic user-item interaction graph (DIG) is a directed bipartite graph $G = (V_u, V_i, A)$. User part V_u contains all the user nodes and item part V_i contains all the item nodes. Directed edge set A consists all the directed edges of the form (h, t, o), where h is the head and t is the tail such that h and t are in different parts, label o indicates the interaction order in terms of h. Earliest interaction is labeled 1 and later is labeled $2, 3, \ldots$. We define the ordered neighbors of node v as the sequence of node reached directly from v sorted by the order of corresponding interactions, that is

$$N_v = (v_1, v_2, \ldots, v_L) \text{ s.t. } \forall 1 \leq l \leq L, (v, v_l, l) \in A, \quad (1)$$

where L is the total number of interactions of v. Each node in N_v is called a neighbor of v. G changes over time by inserting new nodes in V_u or V_i and inserting new interactions into E. We denote G^t as the DIG at time t.

Long-Tail Item Recommendation

In real-world recommender systems, only a small number of items have rich interactions whereas the remaining majority have insufficient interactions (Anderson 2006; Yin et al. 2012), i.e. items lie in the long-tail distribution. Such majority items are called the long-tail items. (Yin et al. 2012) first proposes the long-tail item recommendation problem, we give the DIG-based context-aware recommendation version of this problem as follows:

Given a DIG $G = (V_u, V_i, A)$ and a query instance (u, i) where i lies in long-tail distribution, predict the probability that user u prefers item i.

Model

In this section, we introduce the proposed PIGAT in detail, the architecture of which is illustrated in Figure 2. PIGAT is composed of four parts: (1) an embedding layer that transforms sparse features into dense embedding vectors; (2) a group of confidence embeddings to distinguish the confidence of the interaction neighbors in different positions in the sequence; (3) an interactive embedding generator that generate both the interactive head embeddings and adaptive interaction embeddings; (4) final multilayer perceptron (MLP) layer that predicts the probability that the user prefers the item. In the rest of this section, we will elaborate these three parts.

Feature Representation

In PIGAT, the data we used consist of four groups of categorical features: User Profile, Item Profile, User Neighbor Sequence, and Item Neighbor Sequence. User Neighbor Sequence contains the sequence of item profiles corresponding to the ordered neighbors of the user and Item Neighbor Sequence contains the sequence of user IDs corresponding to the ordered neighbors of the item, where ordered neighbors are the nodes directly reached from the user/item in DIG in the order of corresponding interaction's order as defined in Equation 1. Each group of categorical features is represented by a high-dimensional sparse binary feature via one-hot embedding or multi-hot embedding. An example is illustrated as follows:

$$\underbrace{[1,0,\ldots,0]}_{\text{user\_id=0}} \quad \underbrace{[0,1,0,\ldots,0]}_{\text{item\_id=1}} \quad \underbrace{[0,0,1,0,\ldots,0]}_{\text{item\_cate\_id=Comedy}}$$

$$\underbrace{[1,0,1,0,\ldots,0]}_{\text{user\_neighbors=}\{0,\,2\}} \quad \underbrace{[0,1,0,1,0,\ldots,0]}_{\text{item\_neighbors=}\{1,\,3\}}$$

Embedding Layer

In embedding layer, high-dimensional sparse features are transformed into low-dimensional dense vectors by looking up embedding tables. Specifically, we build user embedding table $\mathbf{E}_u = [\mathbf{e}_{u_1}, \mathbf{e}_{u_2}, \ldots, \mathbf{e}_{u_{N_u}}] \in \mathbb{R}^{H_u \times N_u}$ to represent User Profile, where H_u is the embedding size of User Profile, N_u is the total number of categorical features in User Profile and $\mathbf{e}_{u_k} \in \mathbb{R}^{H_u}$ is an embedding vector with dimensional of H_u. Item embedding table $\mathbf{E}_i \in \mathbb{R}^{H_i \times N_i}$ is built in a similar way.

Given an input vector $\mathbf{x} \in \{0,1\}^N$ and embedding table $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N] \in \mathbb{R}^{H \times N}$, the embedding of $\mathbf{x}$ is $\mathbf{e} = [\mathbf{e}_{k_1}, \mathbf{e}_{k_2}, \ldots, \mathbf{e}_{k_n}]$, where $j \in \{k_1, k_2, \ldots, k_n\}$ if and only if $x_j = 1$. In this way, User Profile, Item Profile, User Neighbor Sequence, and Item Neighbor Sequence are represented by the embedding vectors $\mathbf{e}_u \in \mathbb{R}^{n_u H_u}$, $\mathbf{e}_i \in \mathbb{R}^{n_i H_i}$, $\mathbf{e}_{uns} \in \mathbb{R}^{L_u \times n_i H_i}$ and $\mathbf{e}_{ins} \in \mathbb{R}^{L_i \times H_u}$, respectively, where n_u (n_i) is the number of categorical features in User Profile (Item Profile) and L_u (L_i) is the count of neighbors of the user (item). Note that User Neighbor Sequence shares the same embedding table with Item Profile and Item Neighbor Sequence shares the same embedding table with User Profile, which enables PIGAT to integrate the graph information with contextual features.

Confidence Embedding

In recommender systems, recent interactions are usually more reflective of user preference than previous interactions, which should play a more credible role in the recommendation process. To distinguish the confidence of interactions occurring at different time, we introduce the confidence embedding into PIGAT. In this work, we initialize the confidence embedding with the following equation:

$$CE_{(l,i)} = \exp(l - L - 1)\cos((i-1)\pi/H), \qquad (2)$$

where l indicates the order of the interaction, i indicates the index of the unit in the embedding, L is the total number of interactions and H is the dimension of the embedding. Note that for a particular interaction, the confidence embedding of it forms a cosine curve with length π; for a particular embedding index, the embedding value is exponential decay according to the time-reverse order of interactions. Since interactions are represented by the ordered neighbors in our model, then given the input interaction neighbor embedding sequence, the confidence embedding with the same embedding dimension is added to it before computing the attention coefficients.

Our confidence embeddings share the similar idea as positional encodings in NMT task (Gehring et al. 2017; Vaswani et al. 2017), which is to make use of the order of sequence. However, we use exponential scalar to model the decay of the interaction confidence instead of just distinguish the different position in the sequence.

Interactive Embedding Generator

PIGAT aims to capture dynamic and adaptive user interests and the item attractions from the DIG. Thus in contrast with traditional models like (Shan et al. 2016; Cheng et al. 2016) in which embeddings are directly fed into the MLP network, PIGAT uses the interactive embedding generator to generate more effective embeddings before MLP layer.

As shown in Figure 2, interactive embedding generator consists of two components: (1) a pairwise attention layer to capture the interactive relationship between ordered neighbors and the head, and the adaptive relationship between user interests (item attractions) and recommendation context; (2) an integrate layer to generate the final representation of head node features and adaptive interaction embeddings.

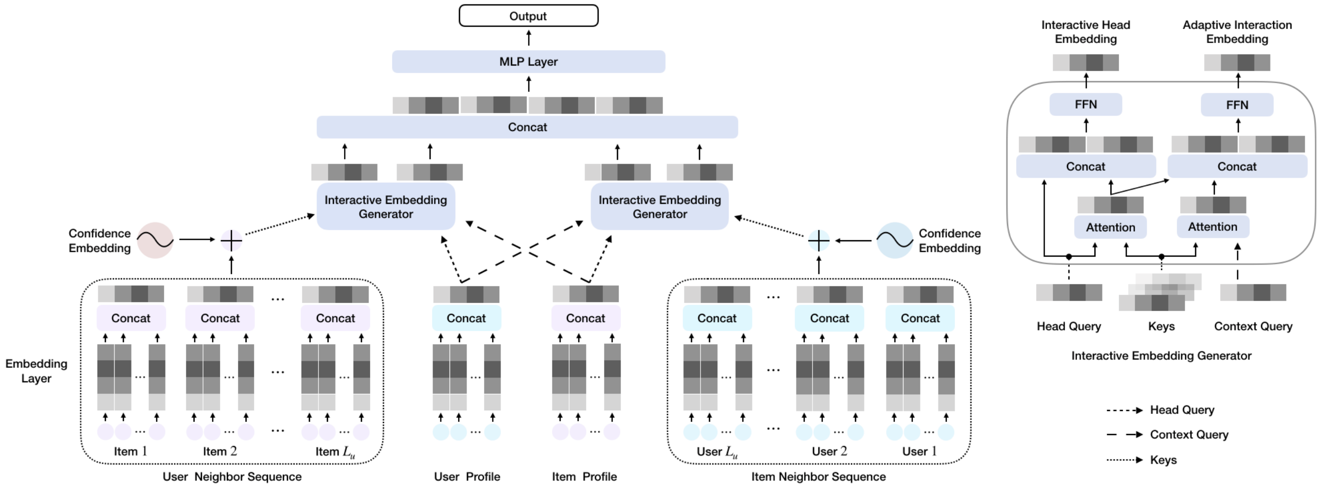


Figure 2: Model Architecture.

Pairwise Attention Layer In recommender systems, user-item interactions reflect user interests and item attractions, which can be used to enhance the primary user/item representation and estimate the relevance to the recommendation context. Based on this fact, we introduce the pairwise attention layer to measure the importance of each interacted neighbor to both the head and the context, and then perform weighted sum pooling to generate interactive embeddings. Mathematically, after the embedding layer as described above, we obtain $\mathbf{e}_u$, $\mathbf{e}_i$, $\mathbf{e}_{uns}$ and $\mathbf{e}_{ins}$ to represent User Profile, Item Profile, User Neighbor Sequence and Item Neighbor Sequence, respectively. We then take these embeddings as the input embeddings of the pairwise attention layer, and compute the attention coefficients

$$\begin{aligned}
a_{ui}^{(l)} &= \text{softmax}(\text{FFN}_{ui}(\mathbf{e}_u, \mathbf{e}_{uns}^{(l)})), \\
a_{ua}^{(l)} &= \text{softmax}(\text{FFN}_{ua}(\mathbf{e}_u, \mathbf{e}_{uns}^{(l)})), \\
a_{ii}^{(l)} &= \text{softmax}(\text{FFN}_{ii}(\mathbf{e}_u, \mathbf{e}_{ins}^{(l)})), \\
a_{ia}^{(l)} &= \text{softmax}(\text{FFN}_{ia}(\mathbf{e}_u, \mathbf{e}_{ins}^{(l)})),
\end{aligned} \tag{3}$$

where $\cdot^{(l)}$ denotes the embedding of the l-th neighbor, $\text{FFN}_{ui}(\cdot)$, $\text{FFN}_{ua}(\cdot)$, $\text{FFN}_{ii}(\cdot)$, and $\text{FFN}_{ua}(\cdot)$ are four independent feed-forward neural networks (FFNs) with the same structure but different weights and the output of each FFN is normalized through the softmax function:

$$\text{softmax}\left(x_i\right) = \frac{\exp\left(x_i\right)}{\sum_j \exp\left(x_j\right)}. \tag{4}$$

a_{ui}, a_{ua}, a_{ii}, a_{ia} are called user interactive weight, user adaptive weight, item interactive weight, item adaptive weight, respectively. Note that for user (item) neighbor sequence, interactive weights characterize the importance of each neighbor to the head, and adaptive weights characterize the importance of each neighbor to the context. Finally, the obtained attention coefficients are used to calculate the interactive embeddings and adaptive embeddings as follows:

$$\begin{aligned}
\mathbf{h}_{ui} &= \sum_{l=1}^{L_u} a_{ui}\mathbf{e}_{uns}^{(l)}, \quad \mathbf{h}_{ua} = \sum_{l=1}^{L_u} a_{ua}\mathbf{e}_{uns}^{(l)}, \\
\mathbf{h}_{ii} &= \sum_{l=1}^{L_i} a_{ii}\mathbf{e}_{ins}^{(l)}, \quad \mathbf{h}_{ia} = \sum_{l=1}^{L_i} a_{ia}\mathbf{e}_{ins}^{(l)},
\end{aligned} \tag{5}$$

where L_u, L_i are the neighbor number as described above.

Integrate Layer After the pairwise attention layer, the outputted interactive embeddings and adaptive embeddings are integrated with original profile embeddings in the integrate layer. Specifically, the interactive embedding and original profile embedding are concatenated, then single-layer FFNs are used to generate the final interactive embeddings, which can be formulated as follows:

$$\begin{aligned}
\mathbf{h}_u &= \text{LeakyReLU}\left(\mathbf{W}_u[\mathbf{e}_u \| \mathbf{h}_{ui}] + \mathbf{b}_u\right), \\
\mathbf{h}_i &= \text{LeakyReLU}\left(\mathbf{W}_i[\mathbf{e}_i \| \mathbf{h}_{ii}] + \mathbf{b}_i\right),
\end{aligned} \tag{6}$$

where $\mathbf{W}_u \in \mathbb{R}^{n_i H_i \times n_i H_i}$, $\mathbf{W}_i \in \mathbb{R}^{n_u H_u \times H_u}$ are the weights of FFNs to generate user interactive embedding and item interactive embedding respectively, $\mathbf{b}_u \in \mathbb{R}^{n_i H_i}$, $\mathbf{b}_i \in \mathbb{R}^{n_u H_u}$ are the bias of FFNs, and $\|$ is the concatenation operation. In addition, LeakyReLU activation is applied to the fully-connected layer. Analogously, the outputted interactive embedding and adaptive embedding are concatenated then fed into the FFNs to generate adaptive interaction embeddings $\mathbf{h}'_u$ and $\mathbf{h}'_i$.

MLP Layer

In MLP layer, the interactive embeddings ($\mathbf{h}_u$, $\mathbf{h}_i$) and adaptive interaction embeddings ($\mathbf{h}'_u$, $\mathbf{h}'_i$) generated by the interactive embedding generator are concatenated, and then fed into the final MLP to predict the probability that user prefers the item.

Loss Function

Given an training instance $(\mathbf{x}, y)$, our target is to maximize the predicted probability $\hat{y} = f(\mathbf{x})$ if $y = 1$, otherwise ($y = 0$) is to minimize $\hat{y}$. Since then, we take the negative log-likelihood function as our loss function, which is defined as follows:

$$L = -\frac{1}{n} \sum_{(\mathbf{x},y)\in D} (y \log f(\mathbf{x}) + (1-y)\log(1-f(\mathbf{x}))), \quad (7)$$

where D is the training set.

Experiments

In this section, we perform experiments on three real-word datasets to evaluate the performance of our proposed PIGAT. We start by introducing the detailed experimental setup, and then presents the experiment results and analysis. Experiments shows that PIGAT outperforms state-of-the-art methods on user preference prediction task.

Experimental Setup

Datasets We conduct experiments on both benchmark datasets and tens-of-millions sized grand challenge datasets to evaluation the effectiveness of our proposed approach.

- **Amazon**[1]. Amazon dataset is a widely used benchmark dataset (He and McAuley 2016; McAuley et al. 2015; Zhou et al. 2018b), which contains product reviews and metadata from Amazon. All users and items in the dataste have at least 5 reviews. Our experiments are conducted on two subsets of Amazon Dataset: Books and Electronics. We select the ID of the reviewer as the User Profile, select the ID and categories of the product as the Item Profile, take the reviewed product as User Neighbors and take the reviewing user as Item Neighbors. Furthermore, we label the instances with overall rating above 3 as positive and the rest as negative.
- **Byte-Recommend**[2]. Byte-Recommend dataset is a large public grand challenge dataset, which contains of tens of thousands of different users and millions of different videos. We use user_id and device_id as User Profile, use item_id and author_id as Item Profile, take the watched videos as User Neighbors and take the watching user as Item Neighbors. We directly use the finish indicator (indicating whether the user finishes watching the video) in the dataset as the label.

The statistics of all above datasets are shown in Table 1. Note that in Byte-Recommend dataset, item density is much lower than other datasets, which leads to more serious long-tail problem.

Competitors We compare our model with the following models to evaluate the performance:

- **FM** (Rendle 2010) Factorization machine (FM) is a classical context-aware recommendation model, which captures the feature interaction through the inner-product.
- **YoutubeNet** (Covington, Adams, and Sargin 2016) YoutubeNet is a classical model following the Embedding&MLP paradigm.
- **DeepFM** (Guo et al. 2017) DeepFM models low-order relationship and high-order relationship between features simultaneously and shares embeddings between two components.
- **DIN** (Zhou et al. 2018b) DIN is a state-of-art model for context-aware recommendation, which use attention mechanism to learn the relationship between user behaviors and the candidate item. We take the user interactive neighbors as user behaviors and take the item interactive neighbors as common context features.
- **GC-MC** (van den Berg, Kipf, and Welling 2017) GC-MC is a state-of-art GCN-based model. We take the user profile and the item profile as side information.

For FM, we conduct experiments with and without interaction neighbor sequence (refer as FM$-$) and for YoutubeNet, we conduct experiments with and without item interaction neighbor (refer as YoutubeNet$-$) to verify the effectiveness of the interaction neighbor sequence. In FM and DeepFM, the interaction neighbor sequence are treated as undifferentiated sparse features. In YoutubeNet, the interaction neighbor sequence are tuned into fixed-length embedding through average pooling operation.

Evaluation Protocols We split each dataset into training (80%), validation (10%), and test (10%) sets according to the timeline, where the validation set is to tune hyper-parameters and performance comparisons are taken on the test set. The task is to predict the label in each dataset. We only use last 10 interactions (with the largest 10 labels) of users or items with no more than 10 interactions for all models and all datasets. To be fair, we implement all models in Pytorch and use Adam optimizer (Kingma and Ba 2015) to optimize all models. The embedding size is fixed to 64 on Byte-Recommend dataset and 128 on all other datasets for all models. For YoutubeNet, DeepFM, DIN and PIGAT, the MLP layer is set to contain three layers with hidden size 80, 40, 1, respectively. The batch size is fixed to 4096. We set the learning rate decayed by constant rate every 1 or 2 epoch(s), the learning rate is selected in $\{10^{-5}, 5\times10^{-5}, 10^{-4}, 5\times10^{-4}, 10^{-3}\}$, and the decay rate is selected in $\{0.1, 0.2, 0.5, 1.0\}$. To overcome the overfitting problem, we apply L_2 regularization with the coefficient in $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ and dropout (Srivastava et al. 2014) with the ratio in $\{0.0, 0.1, 0.2, 0.5\}$ is applied to the input of the MLP layer.

Metrics Area Under ROC Curve (AUC) measures the ranking ability of the model (Fawcett 2006), which is a widely used metric in context-aware recommendation. It is defines as follows:

$$\text{AUC} = \frac{1}{m^+m^-} \sum_{\mathbf{x}^+\in D^+} \sum_{\mathbf{x}^-\in D^-} R\left(f(\mathbf{x}^+), f(\mathbf{x}^-)\right), \quad (8)$$

where D^+ is the set of all positive instances with size m^+, D^- is the set of all negative instances with size m^-, $f(\cdot)$ is

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

[2] https://biendata.com/competition/icmechallenge2019/data/

Table 1: Statistics of datasets.

Dataset	Instances	Users	Items	avg. # of Users	avg. # of Items
Electronics	1,689,188	192,403	63,001	8.8	26.8
Books	8,898,041	603,668	367,982	14.7	24.2
Byte-Recommend	19,622,340	73,974	4,122,689	277.5	5.3

the prediction model and $R : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a function to compute the ranking score of the input pair, which is defined as follows:

$$R\left(x_1, x_2\right) = \mathbb{I}\left(x_1 > x_2\right) + \frac{1}{2}\mathbb{I}\left(x_1 = x_2\right), \qquad (9)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Performance Comparison Results

Overall Comparison Table 2 shows the AUC score on the test set, from which we have the following observations:

- YoutubeNet performs better when using item neighbor sequence on all three datasets, FM performs better when using interaction neighbor sequences on two larger datasets, which verifies the importance of introducing user-item interactions into recommendation models.
- All the deep models achieve better performance than FM, which indicates that using inner-product to capture only the low-order feature interactions is insufficient. DeepFM outperforms YoutubeNet on Amazon Electronics and Byte-Recommend but underperforms on Amazon Books, which implies that it's insufficient to regard the interaction neighbor sequence same as other features, although complex feature relationship is taken into consideration. DIN improves the performance significantly owe to its specially designed structure to extract user interests. GC-MC captures the relationship between user/item and neighbors in interaction graph, which demonstrates the final representation of user/item features and might limit the model performance for inadequate expressive of User Profile and Item Profile.
- Our model achieves best performance on all datasets in terms of AUC score, especially on Byte-Recommend Dataset with a large volume of long-tail items. It learns both the importance to the head and the adaptive relationship between interactive neighbors and recommendation context, thereby obtaining more effective interactive embeddings to represent user preference and item attractions.

Long-Tail Recommendation Comparison To verify the effectiveness of PIGAT to do long-tail recommendation, we calculate the AUC score on the long-tail subsets which contain items with no more than k neighbors in the training set. The k is called the long-tail threshold and is chosen from $\{3, 5, 10\}$. Results are shown in Figure 3, we omit the result for FM since other models beat it significantly.

Table 2: Performance Coparison on all Datasets.

Model	Books	Electronics	Byte-Rec
FM−	0.6596	0.6279	0.6822
FM	0.6763	0.6183	0.6979
YoutubeNet−	0.7643	0.7004	0.7310
YoutubeNet	0.7677	0.7014	0.7385
DeepFM	0.7666	0.7016	0.7391
DIN	0.7684	0.7027	0.7392
GC-MC	0.7668	0.7025	0.7387
PIGAT	**0.7694**	**0.7033**	**0.7422**

It can be observed that the performance of different models shows a similar trend on each dataset. PIGAT achieves the best performance on all three datasets, especially on Byte-Recommend dataset which contains numerous long-tail items. This verifies that the embedding representations generated by our interaction embedding generator improve the long-tail item recommendation indeed. What's more, PIGAT improves the performance more significantly when long-tail threshold $k = 3$ than larger thresholds, which further verified the ability of PIGAT to generate expressive enough representations for items with extremely rare interactions.

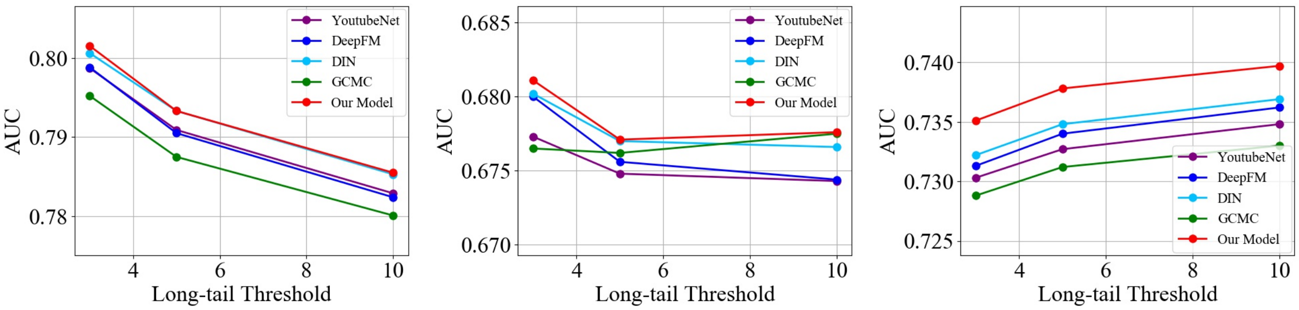
Study of our Model

Effect of Dynamic Interaction Graph Features To verify the effectiveness of dynamic interaction graph features, we replace the dynamic edges by static edges representing the latest 10 interactions in the training set. Table 3 summarizes the results. It shows that recommendation performance decreases a lot by using the static interaction graph. This might be caused by the mismatch between the static interactions and dynamic user interests. This verifies the necessity to introduce dynamic interactions into content-aware recommendation systems.

Table 3: Effect of dynamic interaction graph features.

Graph Type	Books	Electronics	Byte-Rec
Static	0.7684	0.6960	0.7234
Dynamic	**0.7694**	**0.7033**	**0.7422**

Effect of Confidence Embedding To study the influence of confidence embedding to recommendation performance, we conduct experiments on following variants by replacing our confidence embedding to other structures: removing confidence embedding which is donated as None, using positional embedding as (Vaswani et al. 2017) which is



(a) AUC on Books (b) AUC on Electronics (c) AUC on Byte-Recommend

Figure 3: Performance comparison under different long-tail threshold.

donated as PE, using fixed confidence embedding as Equation (2) which is donated as FCE, using random initialed learned embedding which is denoted as RCE. Table 4 summaries the results, from which we observed that

- CE achieves best performance, which verifies the effectiveness of our confidence embedding.
- Using fixed confidence embedding decreases the performance obviously, which implies the variety of interaction confidence might not be able to be generalized by the same rule.
- FCE outperforms PE in most cases, which indicates our confidence embedding is more suitable to model interaction confidence than positional embedding.
- RCE sometimes outperforms model with out confidence embedding, but generally underperforms CE, which further verifies the necessary to introduce special designed confidence embedding.

Table 4: Effect of confidence embedding.

Model	Books	Electronics	Byte-Rec
None	0.7685	0.7009	0.7416
PE	0.7623	0.6960	0.7407
FCE	0.7671	0.6991	0.7414
RCE	0.7683	0.7010	0.7419
CE	**0.7694**	**0.7033**	**0.7422**

Effect of Attention Function To study how attention functions influence our architecture, we conduct experiments on variants using dot-product, scaled dot-product (Vaswani et al. 2017), and FFN as attention functions. As shown in Table 5, using FFN generally outperforms using (scaled) dot-product attention and using FFN-3 achieves best performance, which implies the existence of higher-order relationship between graph node representations and context features.

Table 5: Effect of attention function. Dot-product denotes model using dot-product attention, Dot-product-S denotes model using scaled dot-product attention, FFN-n denotes model using n layer(s) FFN as attention function.

Model	Books	Electronics	Byte-Rec
Dot-product	0.7683	0.7007	0.7413
Dot-product-S	0.7682	0.7010	0.7417
FFN-1	0.7683	0.7020	0.7417
FFN-2	0.7682	0.7019	0.7419
FFN-3	**0.7694**	**0.7033**	**0.7422**

Further Discussion Jointly analyze Table 2, Table 4, Table 5 and Figure 3 we have observed that without confidence embedding, our model already outperforms other model in Books and Byte-Recommend, but confidence embedding leads to further improvements of the performance. For Byte-Recommend, replacing either the confidence embedding or the attention function does not affect the model to achieve best performance, which indicates the superiority of the interactive embedding generator framework for long-tail item recommendation.

Conclusion

In this work, we propose a GNN-based context-aware recommendation model PIGAT, which follows attention mechanism to generate expressive representation of user-item interactions and interactive user/item representations. To capture dynamic user interests, we use the dynamic user-item interaction graph rather than a static graph. The key of PIGAT is to consider two types of the importance of each interaction neighbor: the importance to the head and the importance to the candidate. With the previous one, a more expressive head node representation can be generated. With both of them, the relationship between user interests, item attractions and recommendation context can be captured. We further apply the confidence embeddings to model the variety of interaction confidence. Experiments on three datasets show that the above considerations improve the model performance significantly especially for long-tail item recommendation.

References

[Anderson 2006] Anderson, C. 2006. *The long tail: Why the future of business is selling less of more*. Hachette Books.

[Bahdanau, Cho, and Bengio 2015] Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[Cheng et al. 2016] Cheng, H.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; Anil, R.; Haque, Z.; Hong, L.; Jain, V.; Liu, X.; and Shah, H. 2016. Wide & deep learning for recommender systems. In *DLRS@RecSys*, 7–10.

[Covington, Adams, and Sargin 2016] Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *RecSys*, 191–198.

[Fawcett 2006] Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27(8):861–874.

[Feng et al. 2019] Feng, Y.; Lv, F.; Shen, W.; Wang, M.; Sun, F.; Zhu, Y.; and Yang, K. 2019. Deep session interest network for click-through rate prediction. In *IJCAI*, 2301–2307.

[Gehring et al. 2017] Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *ICML*, 1243–1252.

[Guo et al. 2017] Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: A factorization-machine based neural network for CTR prediction. In *IJCAI*, 1725–1731.

[He and McAuley 2016] He, R., and McAuley, J. J. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, 507–517.

[Kingma and Ba 2015] Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[Kipf and Welling 2017] Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[Lian et al. 2018] Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; and Sun, G. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *KDD*, 1754–1763.

[McAuley et al. 2015] McAuley, J. J.; Targett, C.; Shi, Q.; and van den Hengel, A. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*, 43–52.

[Qu et al. 2016] Qu, Y.; Cai, H.; Ren, K.; Zhang, W.; Yu, Y.; Wen, Y.; and Wang, J. 2016. Product-based neural networks for user response prediction. In *ICDM*, 1149–1154.

[Rendle 2010] Rendle, S. 2010. Factorization machines. In *ICDM*, 995–1000.

[Shan et al. 2016] Shan, Y.; Hoens, T. R.; Jiao, J.; Wang, H.; Yu, D.; and Mao, J. C. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *SIGKDD*, 255–262.

[Srivastava et al. 2014] Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

[van den Berg, Kipf, and Welling 2017] van den Berg, R.; Kipf, T. N.; and Welling, M. 2017. Graph convolutional matrix completion. *CoRR* abs/1706.02263.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*, 5998–6008.

[Velickovic et al. 2018] Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In *ICLR*.

[Wang et al. 2019] Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T. 2019. Neural graph collaborative filtering. In *SIGIR*, 165–174.

[Xiao et al. 2017] Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, 3119–3125.

[Yin et al. 2012] Yin, H.; Cui, B.; Li, J.; Yao, J.; and Chen, C. 2012. Challenging the long tail recommendation. *PVLDB* 5(9):896–907.

[Zhou et al. 2018a] Zhou, C.; Bai, J.; Song, J.; Liu, X.; Zhao, Z.; Chen, X.; and Gao, J. 2018a. Atrank: An attention-based user behavior modeling framework for recommendation. In *AAAI*, 4564–4571.

[Zhou et al. 2018b] Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018b. Deep interest network for click-through rate prediction. In *KDD*, 1059–1068.

[Zhou et al. 2019] Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; and Gai, K. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, 5941–5948.