Neighbour Interaction based Click-Through Rate Prediction via Graph-masked Transformer

Erxue Min[*]
erxue.min@gmail.com
National Centre for Text Mining,
Department of Computer Science,
The University of Manchester
United Kingdom

Yu Rong[†]
Tingyang Xu
Yatao Bian
yu.rong@hotmail.com
tingyangxu@tencent.com
yatao.bian@gmail.com
Tencent AI Lab
China

Da Luo
Kangyi Lin
lodaluo@tencent.com
plancklin@tencent.com
Weixin Open Platform, Tencent
China

Junzhou Huang
jzhuang@uta.edu
Department of Computer Science and
Engineering, University of Texas at
Arlington
United States

Sophia Ananiadou
Sophia.Ananiadou@manchester.ac.uk
National Centre for Text Mining,
Department of Computer Science,
The University of Manchester
United Kingdom

Peilin Zhao
masonzhao@tencent.com
Tencent AI Lab
China

ABSTRACT

Click-Through Rate (CTR) prediction, which aims to estimate the probability that a user will click an item, is an essential component of online advertising. Existing methods mainly attempt to mine user interests from users' historical behaviours, which contain users' directly interacted items. Although these methods have made great progress, they are often limited by the recommender system's direct exposure and inactive interactions, and thus fail to mine all potential user interests. To tackle these problems, we propose Neighbor-Interaction based CTR prediction (NI-CTR), which considers this task under a Heterogeneous Information Network (HIN) setting. In short, Neighbor-Interaction based CTR prediction involves the local neighborhood of the target user-item pair in the HIN to predict their linkage. In order to guide the representation learning of the local neighbourhood, we further consider different kinds of interactions among the local neighborhood nodes from both explicit and implicit perspective, and propose a novel Graph-Masked Transformer (GMT) to effectively incorporates these kinds of interactions to produce highly representative embeddings for the target user-item pair. Moreover, in order to improve model robustness against neighbour sampling, we enforce a consistency regularization loss over the neighbourhood embedding. We conduct extensive experiments on two real-world datasets with millions of instances and the experimental results show that our proposed method outperforms state-of-the-art CTR models significantly. Meanwhile, the comprehensive ablation studies verify the effectiveness of every component of our model. Furthermore, we have deployed this framework on the WeChat Official Account Platform with billions of users. The online A/B tests demonstrate an average CTR improvement of 21.9% against all online baselines.

[*]Work done during Erxue's internship at Tencent AI Lab.
[†]Corresponding author.

1 INTRODUCTION

In many applications such as online advertising and product search, Click-Through Rate (CTR) is a key indicator in business valuation, which measures the probability of a user clicking or interacting with a candidate item. For applications with a large user base, even a small improvement on CTR can potentially contribute to a large increase in the overall revenue. However, achieving accurate CTR prediction remains a great challenge. This is due to the fact that the

data in CTR prediction problems are usually of large scale and high sparsity, involving many categorical features of different fields.

Typically, the data of CTR prediction is represented as high-dimensional and sparse categorical feature groups. To discover the potential click-through relation between user and item, the most popular learning paradigm is to firstly use an embedding layer to transfer the sparse user/item features to a low-dimensional dense embedding, and then construct the feature fusion & learning models to encode the user preferences, item characteristic or their interactions. Typical models include Wide&Deep [2], DeepFM [8], xDeepFM [18], AFM [34], DeepMCP [21] and so on. However, this learning paradigm treats the sparse categorical feature equally and ignores the intrinsic structures among them, e.g., the sequential order of historical behaviors.

Recently, several studies in user interests modeling [7, 19, 20, 38, 39] emphasize on the sequential structure of user behaviour features. They model the historical items of users as sequences and exploit the sequence modeling methods such as LSTM [12], GRU [3] and multi-head attention [28] to effectively model the user preference. Typical methods include DIN [39], DIEN [38], DSIN [7], SDM [19] and DMR [20], etc. Although existing methods for CTR prediction have achieved significant progress, the above methods only focus on mining the interaction between the candidate item and the user's historical behaviours, which suffers from two limitations: On the one hand, user behaviours might be sparse for inactive users, which rise a cold-start problem and impede the quality of representation. On the other hand, due to the limitation of recommender system's exposure, the direct associated items of a user are not exhaustive enough to reflect all his/her potential interests.

To tackle these limitations, we propose Neighbour-Interaction based CTR (NI-CTR) prediction , which extends the prediction of a candidate user-item pair to their local neighbourhoods in a predefined Heterogeneous Information Network (HIN). Specifically, we construct the HIN based on entities and relations which are associated with users and target items in the CTR prediction. Take video recommendation on WeChat Official Account as an example. In this task, we are interested in the CTR of videos. However, many additional information, such as users' click history of articles or news, the subscribe relation between users and Official Accounts, and the publish relation between official accounts and contents (videos/articles) can reflect the users' clicking preference and contribute important clues for inference. To make use of such rich side information, we construct the HIN based on these entities and relations. Afterwards, we leverage graph sampling methods to retrieve neighbours of both the target user and item in the HIN and integrate them to construct a merged local neighbourhood. In order to effectively learn the user-item clicking preference from the local neighborhood nodes, we consider a wider range of interaction types between those nodes from the explicit and implicit perspectives, and propose to construct the four kinds of interaction graphs: 1) Induced graph edges in the HIN for modeling natural interaction, 2) Similarity graph for modeling node feature similarities, 3) Cross neighbourhood graph for capturing the interaction across user neighbours and item neighbours, and 4) Complete graph for modeling interaction between any nodes in the neighbourhood.

To better encode the information from those interaction graphs, we propose a novel Graph-masked Transformer (GMT) architecture
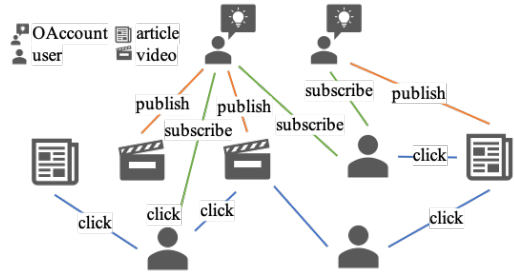

Figure 1: An illustration of the constructed HIN. It contains four kinds of nodes (OAccount, article, user and video) and three kinds of edges (click, publish and subscribe).

to encode the neighbourhood, which can flexibly involve structural priors via a masking mechanism. In this way, the Transformer network is regularized by the graph priors and capable to learn more distinctive representations for the neighbourhood. Besides, in order to reduce the noise introduced by sampling, we enforce a consistency regularization on the neighbourhood embedding, to make neighbourhood embeddings of an identical user-item pair similar. Note that our neighbour interaction method is different from conventional graph-based methodologies such as HetGNN [36] or HAN [32], where graph neural network architectures such as GCN [17] are exploited to compress features of neighbouring nodes in the graph into single embedding vector before making a prediction. In our case, both explicit and implicit interactions amongst neighbours of the user and item are fully captured, which mitigates the early summarization issue as introduced in [22]. Moreover, the deep architecture of Transformer network enhances the feature interaction and improves the feature extraction capability. Furthermore, the heterogeneous graph setting enables the model to utilize cross-domain information, which can be particularly useful when direct interactions between users and items of the target type are scarce, somehow mitigating the Cold-start issue, which is verified in our experiments. To sum up, our contributions to this paper can be concluded as follows:

- We propose to exploit the neighbourhood of the target user-item pair in a HIN to assist the CTR prediction. Four types of interaction graphs are proposed to describe both explicit and implicit relations among the neighbours.
- We propose a novel Graph-Masked Transformer (GMT), which flexibly encodes topological priors into self-attention via a simple but effective graph masking mechanism.
- We propose a consistency regularization loss over the neighbourhood representation to alleviate the uncertainty of graph sampling, and thus improve the robustness of the model.
- We evaluate our method on both public and industrial datasets, demonstrating significant improvements of our methods over state-of-the-art methods in both CTR prediction and graph modeling. Furthermore, we have deployed our framework in the video service of WeChat Official Account Platform, and Online A/B tests also show that it outperforms existing online baselines by 21.9%.

2 PRELIMINARIES

2.1 CTR Data Modeling

Data in industrial CTR prediction tasks is mostly in a multi-group categorical form. For example, [gender=Male, visited_categories={Sports}, visited_tag={Football, basketball}], which are normally transformed into high-dimensional sparse binary features via encoding. Formally, given a user or an item , its feature vector can be represented as $[\mathbf{f}_1, \cdots, \mathbf{f}_k]$, where $\mathbf{f}_i \in \mathbb{R}^{d_{f_i}}$ is the encoding vector in the i-th feature group $\mathcal{F}_i$, k is the total number of fields. Mathematically, we adopt the one/multi-hot representation with dimension d_{f_i} to encode the categorical feature group, where d_{f_i} is the number of unique ids in $\mathcal{F}_i$. $\mathbf{f}_i[j]$ is the j-th element of $\mathbf{f}_i$, $\sum_{j=1}^{d_{f_i}} \mathbf{f}_i[j] = p$, with $p = 1$ refers to one-hot encoding and $p > 1$ to multi-hot encoding. For example, the aforementioned instance with three groups of features are illustrated as:

$$\underbrace{[0, 1]}_{\text{gender=Male}} \quad \underbrace{[0, \ldots, 1, \ldots 0]}_{\text{visited\_categories=\{Sports\}}} \quad \underbrace{[0, \ldots, 1, \ldots, 1, \ldots 0]}_{\text{visited\_tag=\{Football, basketball\}}} \quad .$$

As the categorical features can be extremely high-dimensional and sparse for CTR prediction task, people usually apply an embedding layer which makes a linear transformation on each feature group to generate the low-dimensional dense representations, i.e., $\mathbf{x}_i = \mathbf{W}_i \mathbf{f}_i, \mathbf{W}_i \in \mathbb{R}^{d_{x_i} \times d_{f_i}}$, d_{x_i} is much smaller than d_{f_i}. In this way, we can obtain the corresponding dense feature representation $[\mathbf{x}_1, \cdots, \mathbf{x}_k]$.

The traditional CTR prediction models concern two types of entities: user and item. In this paper, we consider more than two kinds of entities and try to model the CTR problem under heterogeneous graph settings. Assume the node type set is $\mathcal{T}_\mathcal{V}$, the original feature vector of node i which belongs to type $t(i) \in \mathcal{T}_\mathcal{V}$ is denoted as: $\mathbf{f}_{t(i)} = [\mathbf{f}_1^{t(i)}, \cdots \mathbf{f}_{k_{t(i)}}^{t(i)}]$. The corresponding low-dimensional dense feature vector is denoted as $\mathbf{x}_{t(i)} = [\mathbf{x}_1^{t(i)}, \cdots, \mathbf{x}_{k_{t(i)}}^{t(i)}]$.[1] It's worth to note that nodes of different types may share the same feature groups, e.g., different categories of item (e.g., video, article, product) might share the same tag scheme. Given two node types t_a and t_b, we denote $g(t_a, t_b)$ as the indices of the shared feature group of t_a and t_b node types in t_a node type.

2.2 Heterogeneous Graph Construction

In this paper, we consider an undirected Heterogeneous Information Network $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{T}_\mathcal{V}, \mathcal{T}_\mathcal{E})$, where $\mathcal{N}$ is the node set, $\mathcal{E} \in \mathcal{N} \times \mathcal{N}$ is the edge set,$\mathcal{T}_\mathcal{V}$ is the node type set and $\mathcal{T}_\mathcal{E} \in \mathcal{T}_\mathcal{V} \times \mathcal{T}_\mathcal{V}$ is the edge type set. $\mathcal{N} = \{\mathcal{U}, \mathcal{I}, \mathcal{S}_1, .., \mathcal{S}_{|\mathcal{T}_\mathcal{V}|-2}\}$, where $\mathcal{U}$ is the user set, $\mathcal{I}$ is the item set which we are interested in, $\mathcal{S}_i$ is i-th relevant entity set. We assume that the edge type is decided by start and end node for simplicity. Take the WeChat Video Recommendation Scenario as an example, we construct the HIN based on four types of nodes: user, video, article and official account (OAccount), and five types of edges: user-click-video, user-click-article, user-subscribe-OAccount, OAccount-publish-video and OAccount-publish-article, as illustrated in Figure 1. Our task can also be considered as to

---

[1] For the simplification of notation, the subscript of a vector can either be the node index or the node type.

predict the linkage between a user node and a candidate video node.

2.3 Problem Definition

In this section, we formulate the CTR prediction task with necessary notations. There are a set of M users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$, a set of N items $\mathcal{I} = \{v_1, v_2, \ldots, v_N\}$. The user-item interactions are denoted as a matrix $\mathcal{Y} \in \mathcal{R}^{M \times N}$, where $y_{uv} = 1$ denotes user u clicks item v before, otherwise $y_{uv} = 0$. Given the task-associated HIN $\mathcal{G}(\mathcal{N}, \mathcal{E})$ as described in Section 2.2, we can sample a batch of neighbouring nodes $\mathcal{N}_{uv} \in \mathcal{N}$ for each candidate user u and item v pair. Each node $r \in \mathcal{N}$ is associated with a feature vector as described in Section 2.1, so we denote the feature vector sets of the sampled $\mathcal{N}_{uv}$ as $\mathcal{F}_{uv}$. Besides, we denote context features (e.g., time, matching method, matching score) as $\mathbf{C}$. Therefore, one instance can be represented as:

$$\{\mathcal{F}_{uv}, \mathbf{C}\}. \tag{1}$$

The goal of NI-CTR prediction is to *predict the probability that user u will click item v based on the neighbourhood and context features.*

3 METHODOLOGY

In this section, we describe our framework to solve the NI-CTR prediction task. Figure 2 shows the overall framework: NI-CTR. It contains four main components: 1) Neighbour sampling in HIN, 2) Interaction graph construction on sampled nodes, 3) Graph-masked Transformer which encodes the neighbourhoods, 4) Loss and Optimization. In the following sections, we will elaborate on the details of each component.

3.1 Neighbour Sampling in HIN

For each node r, there exist some relevant nodes in the pre-defined graph that may enrich its representation. Considering that the HIN sampling scenario in large-scale service and each node can be associated with rich features. We have the following requirements: 1) We should sample closest nodes as more as possible, because intuitively close nodes (e.g., one-hop neighbours) usually contains most relevant features/information, 2) We should sample nodes for each type in a pre-defined budget to limit the computational complexity, 3) We hope to sample nodes which have the most interactions (edges) with other nodes to maximize topological information. To balance these requirements, we develop a simple but experimentally effective algorithm named Greedy Heterogeneous Neighbouring Sampling (GHNSampling). GHNSampling iteratively samples a list of nodes for a target node r from one hop to further. Let $\{s_k\}_{k=1}^{|\mathcal{T}_\mathcal{V}|}$ denotes the budget sampling sizes for each node type, C_k denotes the neighbours of type k we have already sampled. GHNSampling greedily retrieves nodes from 1-hop to further until meeting the budget. In l-th hop, we retrieve all the neighbours of nodes in $(l-1)$-th hop as $\mathcal{B}^l$, with $\mathcal{B}_k^l \subset \mathcal{B}^l$ as retrieved nodes of type k. For node $t \in \mathcal{B}^l$, we calculate the number of nodes it connects in the sampled node set C as $f_t = |\{s|(s, t) \in \mathcal{E}, s \in C\}|$. If $|\mathcal{B}_k^l| > s_k - |C_k|$, we sample $s_k - |C_k|$ nodes from $\mathcal{B}_k^l$ with the probability proportional to f_t. We iteratively run the steps until budges of all node types are met. This practice aims to retrieve nodes which have the most interactions with the target node to obtain topological information
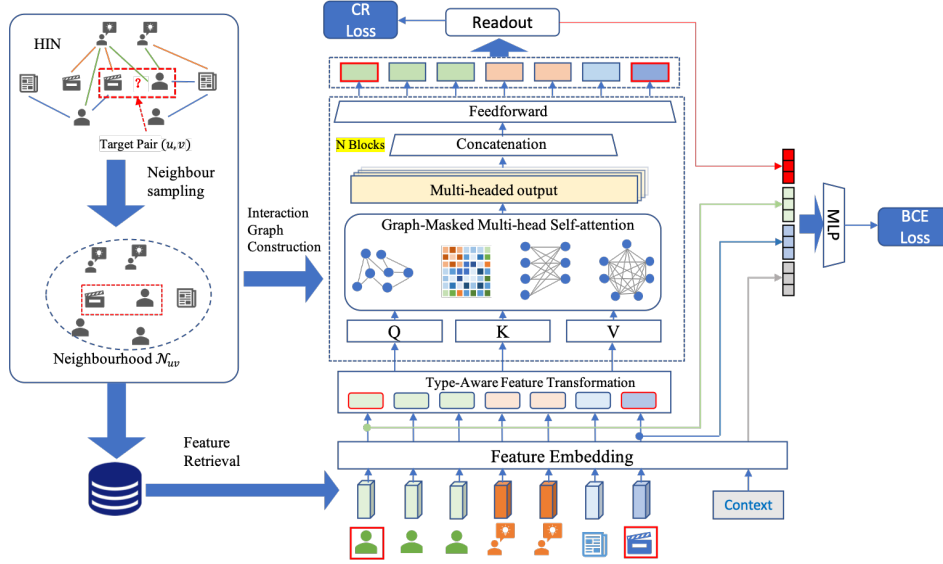
Figure 2: Overview of NI-CTR. Given a target user-item pair, we first perform neighbour sampling in the HIN to obtain associated neighbours. Then we retrieve the corresponding entity features and construct interaction graphs based on the neighbours. After that, we apply a Graph-Masked Transformer to encode both the feature information and topological information. A binary cross-entropy loss and a consistency regularization loss are combined to optimize the network.

Algorithm 1: GHNSampling

Input: HIN $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T}_{\mathcal{V}}, \mathcal{T}_{\mathcal{E}})$, sampling size for each type $\{s_k\}_{k=1}^{|\mathcal{T}_{\mathcal{V}}|}$, Target node r;

Output: Sampled node set C

Initialization: Initialize the node set C as $\{r\}$, $C^0 = \{r\}$

for l *in* $1, 2, 3, \ldots$ **do**

    $\mathcal{B}^l = \{\}, C^l = \{\}$;

    retrieve all neighbours of nodes in $C^{(l-1)}$ and add to buffer $\mathcal{B}^l$;

    $\mathcal{B}^l = \mathcal{B}^l \setminus C$;

    Count connection count f_t for each distinct node t in $\mathcal{B}^l$;

    for *node type* $k \in |\mathcal{T}_{\mathcal{V}}|$ **do**

        Get $\mathcal{B}_k^l \subset \mathcal{B}^l, C_k \subset C$;

        $n_k = \min(|\mathcal{B}_k^l|, s_k - |C_k|)$;

        randomly sample n_k nodes from $\mathcal{B}_k^l$ with probability proportional to f_t and add to C^l;

    end

    $C = C \cup C^l$;

    if $|C_k| = s_k$ *for any node type* k **then**

        Break

    end

end

Return: C

---

as much as possible. The details of GHNSampling are summarized in Algorithm 1.

3.2 Construction of Local Interaction Graphs

After the neighbour sampling for the target user u and candidate item v, we integrate their neighbours to obtain the neighbourhood of the u-v pair, denoted as $\mathcal{N}_{uv}$ ($u, v \in \mathcal{N}_{uv}$). We associate each node i in $\mathcal{N}_{uv}$ with its original feature vector $\mathbf{f}_i$. A direct solution to represent the sequence of nodes is to apply sophisticated models such as Transformer [28], which consider the nodes in the neighbourhood as a complete graph, and learning representations based on the node features. However, to make the representation more distinctive and informative, in this section, we introduce four types of interaction graphs, as illustrated in Figure 3. The details of them are described as follows:

Induced Subgraph $\mathcal{G}_{uv}^I$: It is straightforward that the edge information in HIN provides important natural relation information among the nodes. Therefore, we retrieve all edges from HIN to generate the induced subgraph $\mathcal{G}_{uv}^I$.

Similarity Subgraph $\mathcal{G}_{uv}^S$: In the induced subgraph $\mathcal{G}_{uv}^I$, only a subset of the categorical feature group which describes the behaviour relations or natural relations between different nodes are utilized to construct the graph. However, the other feature groups, such as item tags, which describes the rich latent semantic connections among nodes are ignored. Although these node similarity relations can be implicitly captured by self-attention mechanism in Transformer, they will be decayed after the stacking of multiple layers, which might impede the performance. Therefore, we define the similarity graph $\mathcal{G}_{uv}^S$ by the node feature similarities in the neighbourhood based on the original features of nodes. We calculate all pairwise similarity scores as follows:

$$\text{sim}(i, j) = \frac{\mathbf{f}_i[g(t(i), t(j))] \cdot \mathbf{f}_j[g(t(j), t(i))]}{\|\mathbf{f}_i[g(t(i), t(j))]\| \cdot \|\mathbf{f}_j[g(t(j), t(i))]\|}, \quad (2)$$

where $t(i)$ and $t(j)$ is the type of i- and j-th nodes. $\mathbf{f}_i$ is the original feature vector of node i, $g(t_a, t_b)$ is the feature group indices as mentioned in Section 2.1. Based on the similarity score, we investigate two approaches to construct the *Similarity Graph* $\mathcal{G}_S$: (1)

Weighted similarity graph. we can directly construct the adjacent matrix $\mathbf{M}_S$ based on similarity scores, i.e., $\mathbf{M}_S[i, j] = \text{sim}(i, j)$. (2) *k-NN similarity graph.* Although $\mathbf{S}$ contains weights of similarities, it can be noisy due to data quality. Therefore, we apply a k-NN algorithm on it to retain only strong signals. Namely, $\mathbf{M}_S[i, j] = 1$ if j-th node is one of the k-nearest neighbours of i-th node.

Cross Neighbourhood Subgraph $\mathcal{G}_{uv}^C$: Although the $\mathcal{G}_{uv}^I$ and $\mathcal{G}_{uv}^S$ capture the natural relations and similarity relations of nodes in the neighbourhood. There are more implicit relations we should consider. Let $\mathcal{N}_u$ and $\mathcal{N}_v$ denote the neighbours of u and v respectively, with $\mathcal{N}_{uv} = \mathcal{N}_u \cup \mathcal{N}_v$, we hope to capture all implicit interactions across the two neighbour set, which is inspired by Bidirectional attention flow for machine comprehension [27]. We ensure $\mathcal{N}_u \cap \mathcal{N}_v = \emptyset$ by assigning overlapped nodes to the set with more connected edges. Afterwards, we generate the cross-neighbour graph $\mathcal{G}_{uv}^C = \{(s, t)|s \in \mathcal{N}_u, t \in \mathcal{N}_v\}$.

Complete Subgraph $\mathcal{G}_{uv}^P$: In this graph, we do not impose any structural prior and give the model the most freedom to learn any implicit correlations between nodes. The adjacent matrix of this graph is $\mathbf{M}_P = \mathbf{1}_{|\mathcal{N}_{uv}| \times |\mathcal{N}_{uv}|}$.

3.3 Graph-masked Transformer for Neighbourhood Interaction&Representation

After the construction of Local Interaction Graph, the neighbourhood of one instance can be represented as follows:

$$\{\mathcal{F}_{uv}, \mathcal{G}_{uv}^I, \mathcal{G}_{uv}^S, \mathcal{G}_{uv}^C, \mathcal{G}_{uv}^P\}. \tag{3}$$

In order to learn a representation from both the node features and topological structure. We propose a novel Graph-masked Transformer (GMT), which basically consists of a Heterogeneous Node Feature Transformation layer, stacked Graph-masked Multi-head Self-attention layers and a readout layer. We will introduce the details of each module in the following section.

3.3.1 Heterogeneous Node Feature Transformation layer. For node i in the neighbourhood $\mathcal{N}_{uv}$, we have its embedded dense feature vector $\mathbf{x}_i$ as described in Section 2.1. Since nodes of different type have different feature groups and thus feature space, we use a type-aware feature transformation layer to embed them into a unified space:

$$\mathbf{h}_i = \text{Linear}^{t(i)}(\mathbf{x}_i). \tag{4}$$

where $t(i)$ is the node type of i-th node and $\text{Linear}^t(\cdot)$ is a linear layer of type t, with different types of layers have different trainable parameters.

3.3.2 Graph-masked Multi-head Self-attention. The key difference between GMT and the original Transformer architecture is in the Multi-head Self-Attention (MSA) layers. Given input sequence $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n\}$, the process of basic Self-attention mechanism can be defined as follows:

$$e_{ij} = \frac{(\mathbf{Q}\mathbf{h}_i)^\top (\mathbf{K}\mathbf{h}_i)}{\sqrt{d}}, \tag{5}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^n \exp(e_{ik})}, \tag{6}$$

(a) Induced Subgraph

(b) Similarity Graph

(c) Cross-Neighbourhood Graph

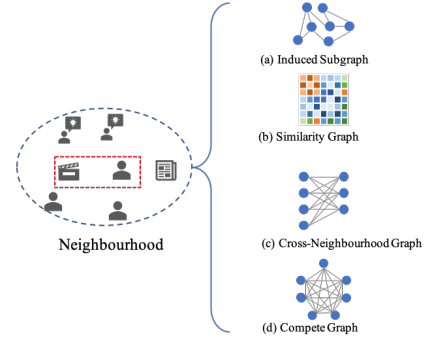(d) Compete Graph

Neighbourhood

Figure 3: Four types of interaction graphs for neighbourhood modeling, which contain natural interactions, feature similarities, cross-neighbourhood interactions and all pairwise interactions.

$$\mathbf{z}_i = \sum_{j=1}^n \alpha_{ij}(\mathbf{V}\mathbf{h}_i), \tag{7}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are trainable parameter matrices, d is the dimension of h_i, $n = |\mathcal{N}_{uv}|$ is the number of neighbours. In a MSA layer, we have H attention heads to implicitly attend to information from different representation subspaces of different nodes. In our model, we attempt to use a *graph-masking mechanism* to enforce the heads explicitly attend to different subspaces with graph priors. Specifically, we modify the calculation of the unnormalized attention score e_{ij} as follows:

$$e_{ij} = f_m\left(\frac{(\mathbf{Q}\mathbf{h}_i)^\top (\mathbf{K}\mathbf{h}_i)}{\sqrt{d}}, \mathbf{M}_{ij}\right), \tag{8}$$

where $\mathbf{M}$ is the adjacent matrix of the prior graph, and $f_m(\cdot)$ is the masking function:

$$f_m(x, \lambda) = \begin{cases} \lambda x & \lambda \neq 0 \\ -\infty & \lambda = 0. \end{cases} \tag{9}$$

This simple yet efficacious way enables the attention calculation aware of the structural priors. Given the four types of interaction graphs described in Section 3.2, we group heads into four sets and apply the graph masking with corresponding adjacent matrix. Now the output representation of i-th node $\mathbf{h}_i'$ is computed as follows:

$$\mathbf{h}_i' = \text{FFN}(\mathbf{W}^O \text{Concat}(\mathbf{z}_i^1, \cdots, \mathbf{z}_i^H)), \tag{10}$$

where $\mathbf{W}^O$ is the parameter matrix, $\text{FFN}(\cdot)$ is a two-layer feed forward layer with layer normalization [1] and residual connection [11]. With our Multi-head Graph-masked mechanism, we incorporate various graph priors into the Transformer architecture, which significantly extends the model expressivity. After stacking multiple Graph-masked MSA layers, we have the final representation of nodes in the neighbourhood as: $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{|\mathcal{N}_{uv}|}\}$.

3.3.3 Readout Layer. To obtain a fixed-sized representation vector of the neighbourhood, we use a readout function:

$$\mathbf{g}_{uv} = \text{Readout}(\mathbf{Z}), \tag{11}$$

in this paper, we simply use a mean pooling function to obtain the final neighbourhood embedding, i.e., $\mathbf{g}_{uv} = \frac{1}{|\mathcal{N}_{uv}|} \sum_{v_i \in \mathcal{N}_{uv}} \mathbf{z}_i$

3.4 Classification and Optimization

In this section, we introduce how to obtain the final prediction score and how to optimize the whole model. Although we have obtained neighbourhood embedding of the target u-v pair, the feature information of the two targets: u and v, despite being most valuable, might be somehow decayed after multiple layers of interaction with neighbours. To highlight the features of the two target nodes, we concatenate their initial dense embeddings $\mathbf{x}_u, \mathbf{x}_v$, with the neighbourhood embedding $\mathbf{g}_{uv}$ and the context features $\mathbf{C}$. So the final embedding of the data instance is:

$$\mathbf{z}^o = \text{Concat}(\mathbf{g}_{uv}, \mathbf{x}_u, \mathbf{x}_v, \mathbf{C}). \quad (12)$$

We adopt a MLP (Multi-Layer Perceptron) layer f_{mlp} with parameter θ and a Sigmoid function σ to predict the probability that the user u will click the target item v:

$$\hat{y}_{uv} = \sigma(f_{mlp}(\mathbf{z}^o, \theta)) \quad (13)$$

Intuitively, in different training epochs, the neighbours sampled from the same u-v pair can be different, which means the neighbourhood embedding $\mathbf{g}_{uv}$ and the final CTR $\hat{y}_{uv}$ would be different. Assuming we sample maximum S times for each u-v pair, so the classification loss is the binary cross entropy loss on the training set $\mathcal{D}$:

$$\mathcal{L}_{\text{BCE}} = \frac{1}{S} \sum_{<u,v> \in \mathcal{D}} \sum_{s=1}^{S} (y_{uv}\log\hat{y}_{uv}^s + (1 - y_{uv})\log(1 - \hat{y}_{uv}^s)), \quad (14)$$

where $\mathcal{D}$ is the training set, $y_{uv} \in \{0, 1\}$ is the ground-truth label of the click-through. In order to improve the robustness of models over sampling randomness, we propose a variant of consistency regularization [6], which enforces the model to learn similar embeddings for the neighbourhoods sampled by the same u-v pair. It can be formulated as:

$$\mathcal{L}_{\text{CR}} = \frac{1}{S} \sum_{<u,v> \in \mathcal{D}} \sum_{s=1}^{S} \frac{1}{d_g}||\hat{\mathbf{g}}_{uv}^s - \bar{\mathbf{g}}_{uv}^s|| \quad (15)$$

, where $\bar{\mathbf{g}}_{uv}^s = \frac{1}{S}\sum_{s=1}^{S} \hat{\mathbf{g}}_{uv}^s$, and d_g is dimension of $\hat{\mathbf{g}}_{uv}^s$. The overall loss is obtained as a summation of both losses with coefficient γ:

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \gamma \mathcal{L}_{\text{CR}} \quad (16)$$

.

3.5 Complexity Analysis

Since scalability is an important issue for CTR prediction in real-world industrial applications, we analyze the time complexity of GMT. The self-attention calculation is the main time cost, the computational complexity of which is $O(n^2 \cdot d)$ where d is hidden size of the neural network. In experiments, we found that we can achieve good performance with $n = 100$. Another time cost is the calculation of similarity graph, which is $O(n^2 \cdot d^{\hat{t}})$, where $d^{\hat{t}}$ is the average dimension of shared node features. Since the node features are usually very sparse, it can be efficiently calculated via sparse matrix libraries.

4 EXPERIMENTS

We compare our method with baselines in both offline evaluation and online service. Results in both settings demonstrate the superiority of our methods.

Table 1: Statistics of the WeChat HIN

Node type	Count	Fields	Features[a]
User	728M	75	147572
OAcc	369K	95	187323
Article	74M	26	148284
Video	846K	23	134758

Edge Type	Count	Ave Src Deg	Ave Dst Deg
user-video	998M	1.35	1167.08
user-article	11.3B	15.53	151.25
user-OAcc	33.9B	46.67	9726.76
OAcc-video	50M	1.43	5.91
OAcc-article	74.7M	21.39	1.0

[a]Here we do not count in any entity (user/OAcc/article/video) ids, which would be extremely large.

4.1 Datasets

Since there is no public large-scale CTR dataset that contains rich heterogeneous graph information, we build a new dataset from WeChat Video Recommendation System for CTR prediction. In our system, generally we have four types of nodes: User (U), Video (V), Article (A) and the content provider named Official Account (O), and five types of edges: user-click-video, user-click-article, user-subscribe-OAccount, OAccount-publish-Video and OAccount-publish-Article. We treat the graph as an undirected heterogeneous graph. We use the log data from 14 consecutive days to construct the WeChat HIN, with over 0.8 billion nodes and 46 billion edges. The detailed statistics are listed in Table 1. The dataset contains the log data of two successive days (Day_1 and Day_2) with 20 million display/click logs of 17 million users and 0.5 million videos for each. Logs from Day_1 are for training and logs from Day_2 are for testing. This dataset is denoted as WC_FULL. Besides, we also build a smaller dataset, using the logs of the first 12 hours of Day_1 for training, while those of the left 12 hours for testing, with around 10 million logs for each. This dataset is denoted as WC_SMALL. We would release the anonymized dataset for reproduction in the near future [2]. Meanwhile, we also evaluated our models on Tmall[3], which contains anonymized users' shopping logs in the past 6 months before and on the "Double 11" day. For each user, their clicked items are sorted by the interaction timestamp. We use the logs of a week before the "Double 11" day as the training set, and the logs on the "Double 11" day as the testing set. For each user, we randomly sample 10 non-clicked items to replace the target item as the negative samples.

4.2 Competitors & Metrics

For offline evaluation, we compare our method with four categories of models: Feature Interaction (FI) models, User Interests Modeling (UIM) models, Graph Neural Networks (GNN) models, and Transformer-based models.FI models include DeepFM [8] and xDeepFM [18]. UIM models include DIN [39], DIEN [38] and DMR [20]. GNN models include GAT [29], GraphSAGE [9], RGCN [26], HAN [32] and NIRec [15]. Transformer-based methods include

---

[2]A demo of data instances and the source codes are published in https://github.com/qwerfdsaplking/F2R-HMT.
[3]https://tianchi.aliyun.com/dataset/dataDetail?dataId=42

Table 2: Results on offline datasets

Category	Model	WC_FULL		WC_SMALL		Tmall	
		AUC	Logloss	AUC	Logloss	AUC	Logloss
FI	DeepFM	0.7009	0.2379	0.7022	0.2365	0.9012	0.1999
	xDeepFM	0.7021	0.2370	0.7042	0.2354	0.9023	0.1978
UIM	DIN	0.7042	0.2345	0.7073	0.2320	0.9034	0.1954
	DIEN	0.7043	0.2347	0.7069	0.2334	0.9045	0.1943
	DMR	0.7098	0.2280	0.7089	0.2310	0.9065	0.1926
GNN	GraphSAGE	0.7032	0.2366	0.7056	0.2378	0.9234	0.1789
	GAT	0.7130	0.2214	0.7145	0.2210	0.9245	0.1776
	RGCN	0.7078	0.2289	0.7101	0.2265	0.9201	0.1801
	HAN	0.7015	0.2378	0.7041	0.2399	0.9180	0.1823
	NIRec	0.7149	0.2200	0.7167	0.2197	0.9246	0.1775
Transformer	Transformer	0.7200	0.2174	0.7260	0.2075	0.9339	0.1700
	Graph-Trans	0.7201	0.2175	0.7277	0.2063	0.9321	0.1715
	Graph-BERT	0.7211	0.2165	0.7290	0.2054	0.9345	0.1693
	GMT	**0.7290**	**0.2103**	**0.7360**	**0.2014**	**0.9410**	**0.1603**

Graph-Bert [37] and Graph-Transformer [4]. We adopt two widely-used evaluation metrics: *AUC* and *Logloss* [8], to evaluate the offline performance. *AUC* measures the goodness of assigning positive samples higher scores than randomly chosen negative samples. A higher *AUC* value indicates a better performance. *Logloss* measures the distance between the predicted scores and the ground-truth labels. A lower *Logloss* value implies better performance. For online evaluation, we use several online deployed CTR prediction models as baselines, including DeepFM [8], GBDT[4] [16], DIN [39], DIEN [38], DeepMCP [21] and DMR [20]. We use the click rate to measure the performance of each method.

4.3 Implementation Details

For each target user-video instance, different categories of models have different forms of input features. For FI models, we concatenate all features together as a vector. For UIM models, the inputs are the concatenated user-video features, along with a sequence of user behaviors. In GNN models, the inputs are the two subgraphs of the target nodes, sampled based on their respective graph sampling models, and we use the final representation of the two target nodes for prediction. The inputs of Transformer-based models are the same as GMT, which are the neighbourhoods sampled by GH-Sampling. Since Graph-Transformer and Graph-Bert are designed for one type of edges, we choose the induced subgraph as their graph information. For fair comparison, we set embedding dimension of all models as 100, the batch size as 128. We tune learning rate from {1e-2,1e-3,1e-4,1e-5}, dropout ratio from 0 to 0.9, hidden size of all deep layers from {64, 100, 128}, number of deep layers from {1, 2, 3, 4}, maximum number of nodes in the subgraph from {50, 100, 200, 400}, the regularization balancing coefficient γ from {0.01, 0.1, 1} We conduct a grid search for parameter selection.

4.4 Results on Offline Datasets

Table 2 shows the experimental results on both WeChat dataset and Tmall. From Table 2, we have the following observations:

- Our proposed GMT achieves obviously better performance and beat the other baselines on all datasets. It verifies that the interaction graphs of multiple types and the graph-masked mechanism empower the Transformer architecture to learn more informative representations.

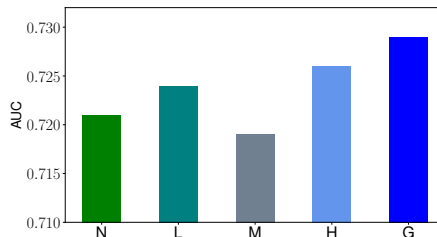[4]GBDT with DeepFM as feature extractor.

Figure 4: Results of graph sampling methods. N: Node-wise sampling; L: Layer-wise sampling; M: Metapath sampling; H: HGSampling; G: GHSampling.

- UIM methods are better than FI methods, demonstrating that user interest mining is useful for representation learning.
- GNN models, especially GAT and RGCN, are generally better than both FI and UIM methods. This is because they utilize the topological information of target nodes, which contains useful auxiliary information and contributes to a better feature interaction.
- The performance of HAN is obviously worse than other graph models. We conjecture that its shallow-layer architecture impairs its capability for node feature interaction.
- We notice that Transformer-based models with our data pipeline are significantly better than other categories of models. It can be attributed to the deep self-attention architecture, which has a more powerful representation capability and learns better feature interaction.

4.5 Ablation Studies

4.5.1 Impacts of four types of interaction graph. In this section, we investigate how each interaction graph in our model influences the final results. Table 3 shows the results of removing or only keeping the masking matrix constructed by the specific interaction graph. We can find that:

- Only keeping a single type of masking matrix achieves obviously inferior performance than the full model (GMT), which means the models cannot learn sufficient structural information of input nodes.
- It is interesting that the model with only cross-subgraph masking performs better than that with fully-connected masking, which implies that the inter-subgraph information aggregation is more important than intra-subgraph information aggregation.
- Meanwhile, the removal of any masking matrix would downgrade the final performance. It demonstrates that each interaction graph contributes to the final results, improving the representation capability of the Transformer Network.

4.5.2 Impacts of consistency regularization. In order to alleviate the noise and uncertainty of neighbour sampling, we enforce a consistency regularization on the embedding of neighbourhood generated by GMT. To verify its effectiveness, we remove this loss and compare it performance against our framework. As we can see in Table 3 (Row 1 and Row 10), the performance of the model declines without the consistency regularization loss. Enforcing the neighbourhood embeddings of an identical *u-v* pair to be similar, the model can be more robust and has a better performance.

Table 3: Ablation results of each module on WC_FULL dataset

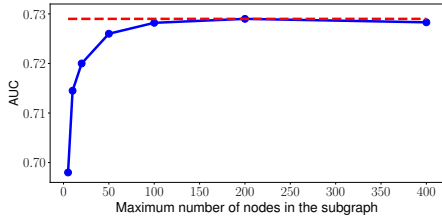| $\mathcal{G}_{uv}^{I}$	$\mathcal{G}_{uv}^{S}$	$\mathcal{G}_{uv}^{C}$	$\mathcal{G}_{uv}^{P}$	CR Loss	AUC	Logloss
		Modules			WC_FULL	
✓	✓	✓	✓	✓	**0.7290**	**0.2103**
✓				✓	0.7180	0.2193
	✓			✓	0.7179	0.2193
		✓		✓	0.7211	0.2154
			✓	✓	0.7203	0.2157
	✓	✓	✓	✓	0.7243	0.2132
✓		✓	✓	✓	0.7252	0.2126
✓	✓		✓	✓	0.7237	0.2149
✓	✓	✓		✓	0.7263	0.2123
✓	✓	✓	✓		0.7274	0.2116
					0.7200	0.2174

Figure 5: Results of different maximum numbers of sampled nodes in the subgraph.
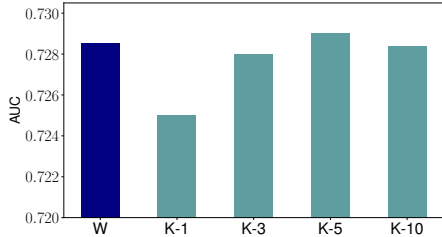

Figure 6: Results of different similarity graphs, where W denotes weighted similarity graph, and K-n denotes k-NN similarity graph with $k = n$.

4.5.3 Impacts of subgraph sampling. In this section, we compare the effects of subgraph sampling on WC_FULL dataset. Firstly, we fix the sampling number n_s of nodes in the subgraph as 200 and compare different graph sampling algorithms. We implement four categories of graph sampling methods: 1) **N**ode-wise sampling [9],with sample number of each node selected form {3, 5, 10}, 2) **L**ayer-wise sampling [14], with layer number selected from 2, 3, 4, 3) **M**etapath sampling [15], with three types of metapaths: UV,UOV,UAUV,UVUV, 4) **H**GSampling [13] with sample depth selected from 2, 3, 4, 5) **G**HSampling.

Figure 4 shows the results of each sampling method with the best parameters. As we can see, layer-wise sampling and HGSampling are better than node-wise sampling and metapath sampling, and the greedy sampling strategy is superior to all methods. It is reasonable, as in such a HIN with rich and meaningful node features, the closest neighbours of a target node contain the most important and relevant information. Node-wise sampling and metapath sampling tend to sample further neighbors than close neighbours, which contains less useful information and introduce extra noise. On the contrary, the GHSampling method samples close neighbours as many as possible and maximizes the inner correlation, thus achieves the
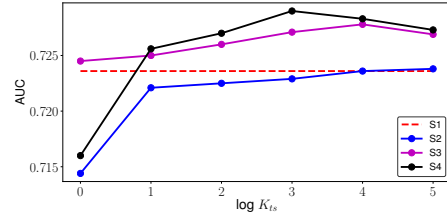

Figure 7: Results of different feature exploitation strategies with varied threshold value K_{ts}.

best performance. Then we analyze the effects of n_s. As illustrated in Figure 5, the performance improves significantly when n_s increases from 5 to 100, and reaches the top when n_s=200. Too large n_s would consume heavy computational costs and impede the results.

4.5.4 Impacts of similarity graph construction. In this section, we compare the effects of the similarity Graph. We compare performance of weighted similarity graph with k-NN graph with k selected from {1, 3, 5, 10}. As illustrated in Figure 6, the performance of the weighted similarity graph is close to that of k-NN similarity graphs with fine-tuned k, and too small k would impair the performance.

Feature exploitation strategies. Moreover, we also conduct an in-depth analysis on four feature exploitation strategies: **S1**) Retain all node features, and do not use the similarity graph masking, **S2**) Remove feature groups with dimension $K > K_{ts}$ from all node features and do not use the similarity graph masking, **S3**) Retain all node features, and use feature groups with dimension $K > K_{ts}$ from node features to calculate similarity masking, **S4**) Remove feature groups with dimension $K > K_{ts}$ from all node features, and use the removed features to calculate the similarity graph, where K_{ts} is a threshold value and we vary K_{ts} from 10^0 to 10^5 to illustrate the variation in performance. As shown in Figure 7, the performance variation of **S2** shows that when $K_{ts} < 10^5$, the more feature groups we use as node features, the better performance we have. However, we can also observe that removing feature groups with a dimension larger than 10^5 in **S2** has slightly better performance than that of **S1**, the baseline which uses all feature groups. It is because extremely sparse features might only bring marginal benefits while consume more model parameters and introduce noise. **S2** shows that using a similarity graph can consistently bring advantages, but calculating the graph based on dense features might introduce noise and impair performance. **S4** generally outperforms **S3**, demonstrating that some sparse features are more suitable to build connections between nodes. Removing them from node inputs improves both effectiveness and efficiency, which is a very practical trick in real service.

4.6 Performance in Cold-start Scenarios

In this part, we conduct experiments on WC_FULL dataset to verify that our model mitigates the Cold-Start problem. Figure 8 shows the performance comparison between DIN and GMT in user cold-start scenarios with respect to different numbers of historical clicked videos for users. The results in the figure illustrated that our method has a more significant relative improvement over DIN when the number of historical videos is smaller, which implies that GMT alleviates the Cold-start issue. The reason is that GMT makes use
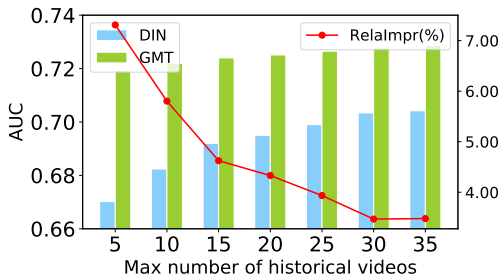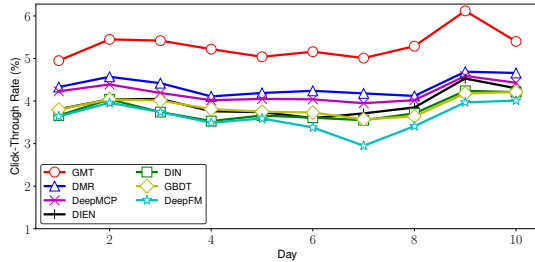
Figure 8: Cold-start analysis result.

Figure 9: Results from Online A/B test during 10 consecutive days. The red curve is our method.

of extra heterogeneous graph information, which helps users to mine potential and implicit connections between users and videos.

4.7 Online Serving & A/B Testing

We conduct online A/B Testing on the Video Recommendation Service of WeChat Official Account Platform during 10 consecutive days. Figure 9 illustrates the results of our method with several baseline models. The results are collected from a consecutive 10 days. As we can see in the Figure, our proposed GMT outperforms other baselines significantly and it improves the best baseline by 21.9% on average. Note that other Top-5 models in the system mostly rely on extra techniques, such as multi-tasking training, pre-hash for item IDs, or node embedding pre-training, which further demonstrates the superiority and robustness of our methods.

Deployment details We save all the node relations into a graph database and update the database daily based on the behavior records of the latest day. We also update the features of each type of node daily based on the profiles of the latest day. We generate training datasets hourly based on the click behaviors in the latest hour. The model is continuously trained on the data of the latest 24 hours with 16 V100 GPUs. The number of training samples for 24 hours is around 40 million. One pass of all samples costs about one hour.

5 RELATED WORK

5.1 CTR Prediction

CTR prediction has been extensively studied for many years. Due to the the high sparsity of input features, it is difficult to achieve good results via directly using raw features, and thus feature interaction modeling becomes a key role in this area [2, 8, 23, 30, 34]. Factorization machines (FM) [23] use a low-dimensional vector to represent each feature field and learns 2-order feature interaction through inner product, achieving a significant improvement over

linear models. Wide&Deep [2] combines a wide linear channel with cross-product and a deep neural network channel to capture feature interaction. DeepFM [8] integrates factorization machines and deep neural networks to learn the second-order crossover of features. xDeepFM [18] propose a novel compressed interaction network (CIN) to generate feature interactions in an explicit fashion and at the vector-wise level, combined with a classical DNN. DCN [30] adopts a multi-layer residual structure to learn higher-order feature representations. AFM [34] uses attention mechanism to automatically learn the weights of cross-features. Apart from learning embedding and interaction on handcrafted features, many works attempts to model user interests from user historical behaviors. Deep Interest Network (DIN) [39] uses attention mechanism to assign different scores to user behaviors to learn the user representation. Deep Interest Evolution Network (DIEN) [38] captures evolving user interest from their historical behaviors on items via a GRU network with attentional update gates. Deep Session Interest Network (DSIN) [7] leverages Bi-LSTM with self-attention layers to model the inter-session and intro-session interests of users. However, although these models try to use powerful network architectures to model different kinds of historical behaviors, they did not make use of multi-source neighbourhood information, which limits their effectiveness.

5.2 Graph Neural Networks for Recommendation

Graph Neural Networks [10, 17, 25, 29] have been widely explored in recommender systems in recent years, owing to their strong capability to model graph information in recommendation. GraphRec [5] makes the first attempt to introduce GNNs to social recommendation by modeling the user-item and user-user interactions as graph data. Wu *etal*. [33] propose a dual graph attention network to collaboratively learn representations for two-fold social effects. KGAT [31] combines user-item graph with knowledge graph and uses graph convolution to obtain the final node representations. Heterogeneous graph Attention Network (HAN) [32] utilizes a semantic-level attention network and a node-level attention network to discriminate the importance of neighbor nodes and node types. HetGNN [36] groups heterogeneous neighbours based on node types and use two modules to aggregate information from them. Despite the progress, it is challenging to directly apply Graph-based recommendation methods to CTR prediction tasks for feature sparsity issues.

5.3 Transformers for Graph Data

There are some attempts to use Transformers in the context of graph-structured data. For example, GTransformer [24] uses a tailed GNN layer as encoder to extract embeddings and feed them into Transformer. Graph Transformer [4] constrain the self-attention mechanism to local neighbourhoods of each node only. Graph-BERT [37] introduces three types of Positional Encoding to embed the node position information to the model. Graphormer [35] utilizes centrality encoding to enhance the node feature and uses spatial encoding along with edge encoding to incorporate structural inductive bias into the attention mechanism. Although these models

have made great progress, they assume that the graphs are homogeneous and only have one type of edges, thus their performances are limited in our setting.

6 CONCLUSION

In this paper, we focus on exploiting the neighbourhood information to improve the performance of CTR prediction. We put CTR prediction into a heterogeneous graph setting and attempt to model the neighbourhood interaction. We propose four types of interaction graphs and design a novel model: Graph-masked Transformer to encode such interactions. Besides, we also design a consistency regularization to enhance the model robustness. Extensive experiments, including detailed ablation studies, verify the effectiveness of the neighbour interaction graph modeling and demonstrates the superiority and the flexibility of the proposed Graph-masked Transformer. We also successfully deploy the whole framework into an industrial scenario with billions of users and items and gain a significant improvement in CTR in a real application.

7 ACKNOWLEDGEMENT

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.
[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[4] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A Generalization of Transformer Networks to Graphs. *arXiv preprint arXiv:2012.09699* (2020).
[5] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.
[6] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph Random Neural Network for Semi-Supervised Learning on Graphs. *arXiv preprint arXiv:2005.11079* (2020).
[7] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).
[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).
[9] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216* (2017).
[10] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. 2019. Cascade-bgnn: Toward efficient self-supervised representation learning on large-scale bipartite graphs. *arXiv preprint arXiv:1906.11994* (2019).
[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[13] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*. 2704–2710.
[14] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems* 31 (2018).
[15] Jiarui Jin, Jiarui Qin, Yuchen Fang, Kounianhua Du, Weinan Zhang, Yong Yu, Zheng Zhang, and Alexander J Smola. 2020. An efficient neighborhood-based

[16] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017), 3146–3154.
[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[18] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.
[19] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2635–2643.
[20] Ze Lyu, Yu Dong, Chengfu Huo, and Weijun Ren. 2020. Deep Match to Rank Model for Personalized Click-Through Rate Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 156–163.
[21] Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Chao Qi, Zhaojie Liu, and Yanlong Du. 2019. Representation learning-assisted click-through rate prediction. *arXiv preprint arXiv:1906.04365* (2019).
[22] Yanru Qu, Ting Bai, Weinan Zhang, Jianyun Nie, and Jian Tang. 2019. An end-to-end neighborhood-based interaction model for knowledge-enhanced recommendation. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–9.
[23] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.
[24] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *arXiv preprint arXiv:2007.02835* (2020).
[25] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Hkx1qkrKPr
[26] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.
[27] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* (2016).
[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
[29] Petar Velivckovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[30] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.
[31] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.
[32] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.
[33] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *The World Wide Web Conference*. 2091–2102.
[34] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).
[35] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Bad for Graph Representation? *arXiv preprint arXiv:2106.05234* (2021).
[36] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.
[37] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140* (2020).
[38] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.

[39] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.