A Graph-Enhanced Click Model for Web Search

Jianghao Lin[1], Weiwen Liu[2], Xinyi Dai[1], Weinan Zhang[1], Shuai Li[1]
Ruiming Tang[2], Xiuqiang He[2], Jianye Hao[2], Yong Yu[1]
[1]Shanghai Jiao Tong University, [2]Huawei Noah's Ark Lab
{chiangel,daixinyi,wnzhang,shuaili8,yyu}@sjtu.edu.cn,{liuweiwen8,tangruiming,hexiuqiang1,haojianye}@huawei.com

ABSTRACT

To better exploit search logs and model users' behavior patterns, numerous click models are proposed to extract users' implicit interaction feedback. Most traditional click models are based on the probabilistic graphical model (PGM) framework, which requires manually designed dependencies and may oversimplify user behaviors. Recently, methods based on neural networks are proposed to improve the prediction accuracy of user behaviors by enhancing the expressive ability and allowing flexible dependencies. However, they still suffer from the data sparsity and cold-start problems. In this paper, we propose a novel graph-enhanced click model (GraphCM) for web search. *Firstly*, we regard each query or document as a vertex, and propose novel homogeneous graph construction methods for queries and documents respectively, to fully exploit both intra-session and inter-session information for the sparsity and cold-start problems. *Secondly*, following the examination hypothesis[1], we separately model the attractiveness estimator and examination predictor to output the attractiveness scores and examination probabilities, where graph neural networks and neighbor interaction techniques are applied to extract the auxiliary information encoded in the pre-constructed homogeneous graphs. *Finally*, we apply combination functions to integrate examination probabilities and attractiveness scores into click predictions. Extensive experiments conducted on three real-world session datasets show that GraphCM not only outperforms the state-of-art models, but also achieves superior performance in addressing the data sparsity and cold-start problems.

CCS CONCEPTS

• **Information systems → Users and interactive retrieval**.

KEYWORDS

Click Model, Web Search, User Modeling, Click Prediction

ACM Reference Format:
Jianghao Lin[1], Weiwen Liu[2], Xinyi Dai[1], Weinan Zhang[1], Shuai Li[1] and Ruiming Tang[2], Xiuqiang He[2], Jianye Hao[2], Yong Yu[1]. 2021. A Graph-Enhanced Click Model for Web Search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462895

[1]Examination hypothesis: a user clicks a document if and only if she examines the document and is attracted by the document.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462895

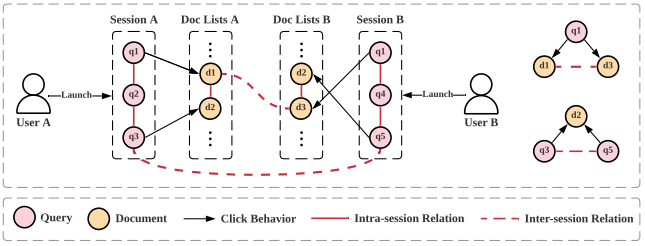

Figure 1: Illustration of intra-session and inter-session information in web search.

1 INTRODUCTION

Understanding users' behavior patterns is key to improving the performance of an information retrieval system. In web search, the ability to summarize users' behavior patterns and precisely simulate user interactions allows search engines to better fulfill users' information needs. To this end, numerous *click models* have been proposed to model users' click behaviors. They serve as click simulators in cases where no real users are available or we prefer not to experiment with real users to avoid hurting user experiences [3]. Besides, click models are also used to estimate the relevance scores for query-document pairs to facilitate document ranking [8, 27].

Earlier click models are based on the *probabilistic graphical model* (PGM) framework [19], where user behaviors are represented as a sequence of observable and hidden states (e.g., clicks, skips, attractiveness, and examinations) [9, 34]. PGM-based click models often require manually designed dependencies between each binary variable, which is likely to oversimplify and therefore overlook some key aspects of user behaviors. To better capture users' behavior patterns and allow flexible dependencies, Borisov [3] proposed a neural click model (NCM), which adopts the *distributed vector representation approach* for user behavior representations, instead of binary variables. While NCM only encodes **query-level** information, the context-aware click model (CACM) [8] utilizes complex structures to incorporate interaction effects among different queries within the same session (i.e., **intra-session** information) and achieves the state-of-art performance among existing click models. However, these click models fail to consider the following two issues.

First, click models generally suffer from the data sparsity problem, i.e., the lack of useful user interaction feedback on query-document pairs. Take *TianGong-ST* dataset[2] for example, the interaction sparsity ratio is 99.9969%.[3] Such sparsity of user interactions

[2]http://www.thuir.cn/tiangong-st/

[3]The interaction sparsity ratio is calculated as $\frac{\#\text{ missing query-document interactions}}{\#\text{ possible query-document pairs}}$.

brings difficulty to the training of click models and leads to poor generalization performance. Although incorporating intra-session information as in CACM may alleviate the sparsity problem, it does not fully utilize the user interactions among queries and documents. In fact, we find that there is rich potential for extracting users' behavior patterns from interactions between queries or documents across different sessions issued by different users (i.e., **inter-session** information). As illustrated in Figure 1, User A and User B launch sessions that consist of different queries towards similar topics . Then queries from different sessions (q_3 and q_5) are likely to be related if they positively interact with the same document d_2, as they share similar user intents. Likewise, documents from different sessions (d_1 and d_3) are related as they are clicked by the same query q_1. Therefore, by better extracting both intra-session and inter-session information, we can achieve superior performance in mitigating the data sparsity problem.

Second, existing click models are vulnerable in the cold-start environments. That is, previous click models generally fail to effectively predict user clicks when there are queries or documents in the test set that never appear in the training set, and thus meeting a dramatic performance decrease during the test phase. For example, in Figure 1, suppose the second query q_4 in Session B is a brand-new query launched by User B during the test phase. Existing click models cannot make reliable predictions for q_4, since little user behavior information is available to form the basis for predictions. Actually, as queries in the same session issued by a user share similar intents towards similar topics, adjacent queries can provide useful reformulation information for the inference and enrich the user interaction representations. For instance, q_4's adjacent queries (q_1 and q_5) in Session B can be regarded as reformulations of q_4, and thereby can be leveraged to predict user behaviors.

To tackle the aforementioned limitations, we propose a novel graph-enhanced click model (GraphCM) in this paper. *Firstly*, we regard each query or document as a vertex and propose novel homogeneous graph construction methods for queries and documents respectively, to fully exploit both intra-session and inter-session information for the sparsity and cold-start problems. *Secondly*, following the examination hypothesis [23], we separately model the attractiveness estimator and examination predictor to output the attractiveness scores and examination probabilities, where graph neural networks and neighbor interaction techniques are applied to extract the auxiliary information encoded in the pre-constructed homogeneous graphs. *Finally*, we combine attractiveness scores and examination probabilities through a combination layer to perform user click prediction. Main contributions of this paper are:

- We propose novel homogeneous graph construction methods for queries and documents respectively, which can fully extract both intra-session and inter-session information.
- We propose a graph-enhanced click model (GraphCM) to exploit structural dependencies among queries and documents according to pre-constructed homogeneous graphs. To the best of our knowledge, this is the first work to apply graph neural networks and neighbor interaction techniques to the field of click models to address the data sparsity and cold-start problems.
- The proposed GraphCM achieves significantly better performance than existing click models in both click prediction and relevance estimation tasks. Besides, extensive empirical studies are conducted to show the ability of GraphCM to tackle the sparsity and cold-start problems.

2 RELATED WORK

2.1 Click Models

To model and simulate user behaviors, numerous click models have been proposed for various application scenarios (e.g., web search, recommendation) [6, 9, 29, 33]. Traditional click models, which are based on the probabilistic graphical model (PGM) framework, treat user behaviors as a sequence of observable and hidden events. They usually incorporate different assumptions on user behaviors to specify how documents and clicks at different positions affect each other. Most PGM-based click models adopt the examination hypothesis [23] where the probability of click are decomposed into the examination probability and the attractiveness score. Different click models study the examination probability differently. A simple click model that follows the examination hypothesis is the position-based model (PBM) [11], which assumes that the examination probability is only related to the displayed positions. The cascade model (CM) [11] assumes that users scan each document in the list from top to bottom until the first click. Yet CM can only handle query sessions with exactly one click. On the basis of CM, user browsing model (UBM) [12], dynamic Bayesian network (DBN) [5], dependent click model (DCM) [15], and click chain model (CCM) [14] have been proposed to overcome this limitation.

As the dependencies in traditional PGM-based click models are designed manually [32], some neural network based approaches have been proposed to get better expressive power and flexible dependencies. The neural click model (NCM) [3] is the first attempt to apply neural networks to click models. NCM treats user behaviors as a sequence of hidden states instead of binary events. The following NN-based click models also adopt this distributed representation framework. The click sequence model (CSM) [4] maintains an encoder-decoder architecture to predict the position sequence of the clicked documents. The context-aware click model (CACM) [8] takes the intra-session information into consideration and separately models the examination probability and the attractiveness score. Our proposed GraphCM can better extract both intra-session and inter-session information by applying graph neural networks and neighbor interaction techniques, resulting in the state-of-art performance compared to existing click models.

2.2 Graph Representation Learning

Graph representation learning investigates low-dimensional representations of graph vertices, while preserving node content and structural information [22, 26, 31]. Earlier graph representation learning methods learn node representations from the graph structure by applying random walks. DeepWalk [20] apply random walks to generate the node context and learn node embeddings. Node2vec [13] further exploits a biased random walk strategy to better capture the local and global structural information. In recent years, graph neural networks are proposed to aggregate information from neighbors and encode structural contexts. GCN [18] applies local graph convolutions for the node classification task. GraphSage [16] performs a non-spectral graph convolution over a fixed

size of sampled neighbors to integrate neighbor features for learning accurate node representations. GAT [18] utilizes a multi-head attention mechanism to increase the model capacity.

GCN, GraphSage and GAT are popular architectures of the general graph neural networks, and can be naturally regarded as plug-in graph representation modules for many supervised tasks [21]. Recently, researchers also deploy graph neural networks in recommender systems to make full use of structural side information and tackle the data sparsity and cold-start problems. For example, PinSage [30] applies graph neural networks to pin-board graph for recommendation. KGAT [28] incorporates user-item graph with knowledge graph to generate finer node representations.

Likewise, general graph neural networks are suitable plug-in modules for click models. To the best of our knowledge, we are the first to introduce graph neural networks to the field of click models, which extracts abundant auxiliary information included in the constructed homogeneous graphs. Therefore, our proposed GraphCM can achieve better performance in addressing the data sparsity and cold-start problems.

3 PROBLEM FORMULATION

The user browsing behaviors in web search is regarded as a series of independent search sessions. A search session $\mathcal{S}$ can be formulated as a sequence of queries $\mathcal{Q}_N = [q_1, ..., q_N]$ submitted by the user. For each query q_i, the search engine returns a ranked list of documents $\mathcal{D}_i = \left[d_{i,1}, d_{i,2}, ..., d_{i,M}\right]$. Each document $d_{i,j}$ has three attributes: the unique URL identifier $u_{i,j}$, the ranking position $p_{i,j}$ and the vertical type[4] $v_{i,j}$. The user browses the ranked lists and may click several documents in the session. We define the click variable $c_{i,j}$ for each document, where $c_{i,j} = 1$ if $d_{i,j}$ is clicked by the user and 0 if not. Then we can define the problem of click model tasks as follows:

Given the user's browsing history $Q = [q_1, q_2, ..., q_n]$, $\mathcal{D} = [d_{1,1}, d_{1,2}, ..., d_{n,m}]$, and $C = \left[c_{1,1}, c_{1,2}, ..., c_{n,m-1}\right]$, for the m-th document $d_{n,m}$ in the n-th query q_n of the session $\mathcal{S}$, we would like to (i) predict whether the document $d_{n,m}$ will be clicked by the user (i.e., the click variable $c_{n,m}$), and (ii) estimate the context-aware relevance between q_n and $d_{n,m}$.

4 MODEL FRAMEWORK

In this section, we introduce the framework of graph-enhanced click model (GraphCM), which is shown in Figure 2.

4.1 Overview of GraphCM

Most click models adopt the examination hypothesis: a user clicks a document $d_{n,m}$ if and only if she examines the document and is attracted by the document [23], which can be formulated as:

$$c_{n,m} = 1 \Leftrightarrow e_{n,m} = 1 \text{ and } a_{n,m} = 1, \tag{1}$$

where $c_{n,m} \in \{0, 1\}$, $e_{n,m} \in \{0, 1\}$, and $a_{n,m} \in \{0, 1\}$ are click, examination, and attractiveness variables, respectively. In this work, for each document $d_{n,m}$, we assume that the user examines it with probability $\mathcal{E}_{n,m}$, and the attractiveness score is $\mathcal{A}_{n,m}$. As shown in Figure 2, GraphCM separately models the attractiveness estimator and examination predictor, and integrates the attractiveness score $\mathcal{A}_{n,m}$ and examination probability $\mathcal{E}_{n,m}$ at click predictor module to output the click probability.

Next, we will first introduce the graph construction methods and general embedding layers, and then elaborate on the details of each module (i.e., attractiveness estimator, examination predictor and click predictor).

4.2 Graph Construction

Thoroughly incorporating intra-session and inter-session information into user behavior modeling is essential for alleviating the data sparsity and cold-start problems. However, it is challenging to fuse complex user behaviors among queries and documents. Moreover, queries and documents have different semantic meanings and should be dealt with differently. To this end, we propose to regard each query or document as a vertex and construct the *query homogeneous graph* and *document homogeneous graph* respectively, to model dependencies in user behaviors. Figure 3 shows an example of these two constructed homogeneous graphs. The query homogeneous graph consists of two kinds of edges:

- **Query Multi-hop Edge**. The edges between queries that click the same document, which denotes the potentially collaborative information between queries.
- **Query-Query Edge**. The edges between each pair of two consecutive queries in the same session, which denotes the reformulation relationships among queries issued by the user.

It is worth noting that the term *reformulation* is different from the traditional IR task *query reformulation*. Here, *reformulation* means that the consecutive queries in the same session issued by a user should share similar intents towards similar topics. Likewise, the document homogeneous graph also contains two kinds of edges:

- **Doc Multi-hop Edge**. The edges between documents that are clicked by the same query, which denotes the potentially collaborative information between documents.
- **Doc-Doc Edge**. The edges between each pair of two consecutive documents in a ranked list, which denotes the similarity relationships among documents.

The graph construction methods above can fully exploit the intra-session and inter-session information (i.e., collaborative, reformulation and similarity information), which helps address the sparsity problem. As for the cold-start problem, a brand-new query or document vertex cannot have multi-hop edges since it has no click interactions available in the training set. As shown in Figure 4, rather than make random guess as in previous click models (no previous click interactions is available), we can aggregate the reformulation or similarity information from the consecutive queries or documents to assist user click prediction, which mitigates the cold-start problem.

4.3 Embedding Layer

GraphCM takes query q, document d, click variable c, vertical type v and ranked position p as inputs. Before the main process of the model, these original ID features are transformed into a high-dimensional sparse features via one-hot encoding. Then we apply embedding layers on the one-hot vectors to map them to

[4]Vertical type means the presentation style of a displayed document (e.g., organic vertical, the illustrated vertical, the encyclopedia vertical) [8].

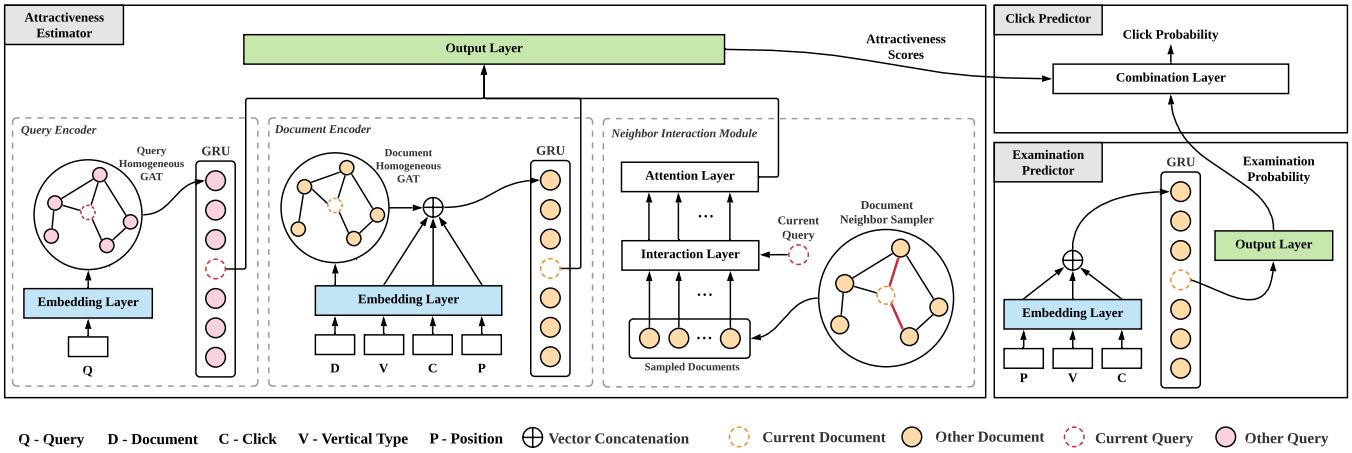

Figure 2: The overall framework of GraphCM. GraphCM consists of an attractiveness estimator and an examination predictor. The attractiveness estimator contains three components (i.e., query encoder, document encoder, and neighbor interaction module), which encode the query context, document context, and query-document interactions to estimate the attractiveness score $\mathcal{A}$. The examination predictor utilizes the session context to predict the examination probability $\mathcal{E}$. GraphCM integrates $\mathcal{A}$ and $\mathcal{E}$ through a combination layer to predict user click behaviors.

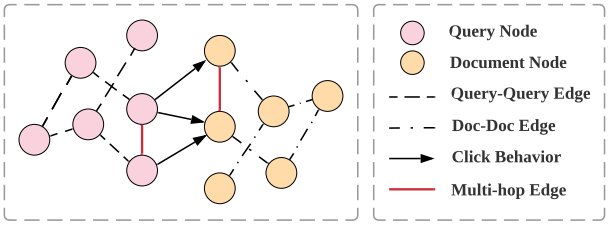

Figure 3: Graph construction methods in GraphCM.

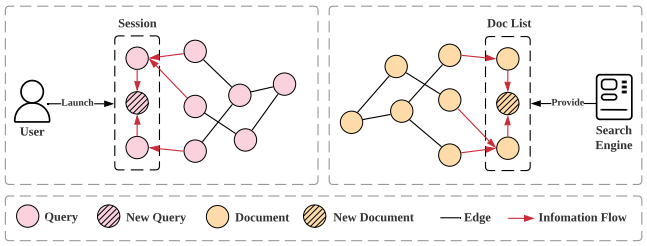

Figure 4: Graph solution for the cold-start problem.

low-dimensional dense embedding vectors:

$$\begin{aligned} \mathbf{v}_q = \mathbf{Emb_q}\left(q\right), \mathbf{v}_d = \mathbf{Emb_d}\left(d\right), \\ \mathbf{v}_c = \mathbf{Emb_c}\left(c\right), \mathbf{v}_v = \mathbf{Emb_v}\left(v\right), \mathbf{v}_p = \mathbf{Emb_p}\left(p\right), \end{aligned} \tag{2}$$

where $\mathbf{Emb}_* \in \mathcal{R}^{N_* \times l_*}$, $* \in \{\mathbf{q}, \mathbf{d}, \mathbf{c}, \mathbf{v}, \mathbf{p}\}$. N_* and l_* denote the input feature size and embedding size. For the ease of presentation, we omit the subscripts of embeddings when there is no ambiguity.

4.4 Attractiveness Estimator

The attractiveness estimator aims to estimate the attractiveness of each document $d_{i,j}$ to the user who issues the query q_i. For the pre-constructed homogeneous graphs, we propose a query encoder and a document encoder respectively to extract intra-session and inter-session information, and generate corresponding query and document context representations. Moreover, we propose a neighbor interaction module to further exploit the high-order neighbor information. Next, we will introduce these three components respectively (i.e., query encoder, document encoder and neighbor interaction module).

4.4.1 Query Encoder. The connectivity in the query homogeneous graph mentioned in Section 4.2 provides collaborative and reformulation information among queries. Therefore, we first apply *graph attention network* [25] (GAT) to aggregate neighbor information, and then generate query context representations via a Gated Recurrent Unit (GRU) [10].

After obtaining query vectors from the embedding layer, we apply GAT based on the query homogeneous graph to model structural dependencies among queries. Since the constructed graph is very large for real-world datasets, it is time consuming and not tractable to directly perform GAT. Therefore, for each query node q_i, we sample a fixed number (e.g., K) of neighbors to form its neighbor set $\mathcal{N}_{q_i}$. It is worth noting that, we add self-loop for every node to ensure that q_i itself can be included in its neighbor set $\mathcal{N}_{q_i}$. Then, we aggregate the information of these neighbors to generate a new embedding $\mathbf{v}'_{q_i}$ for query q_i via a shared asymmetric attention mechanism:

$$\begin{aligned} \alpha_{i,j} &= \text{Softmax}_j(\text{att}_{\text{query}}(\mathbf{v}_{q_i}, \mathbf{v}_{q_j})), \ q_j \in \mathcal{N}_{q_i}, \\ \mathbf{v}'_{q_i} &= \sum\nolimits_{q_j \in \mathcal{N}_{q_i}} \alpha_{i,j} \mathbf{v}_{q_j}. \end{aligned} \tag{3}$$

Here $\text{att}_{\text{query}}$ denotes a linear layer with a LeakyReLU activation function, which performs the asymmetric node-level attention. $\text{att}_{\text{query}}$ is shared for all the nodes in the query homogeneous graph to capture similar connection patterns. Moreover, similar to [24], we extend the proposed attention mechanism to *multi-head attention* to stabilize the learning process of GAT. Specifically, $\mathcal{B}$ independent asymmetric node-level attention mechanisms execute the transformation of Equation 3, and then their features are concatenated,

resulting in the following output feature representations:

$$\mathbf{v}'_{q_i} = \Big\Vert_{b=1}^{\mathcal{B}} \sigma\left(\sum\nolimits_{q_j \in \mathcal{N}_{q_i}} \alpha^b_{i,j} \mathbf{v}_{q_j}\right), \tag{4}$$

where $\|$ denotes vector concatenation, $\alpha^b_{i,j}$ is normalized attention coefficients computed by the b-th attention mechanism $\text{att}^b_{\text{query}}$, σ is the LeakyReLU function. Following [2], to further exploit multi-head attention mechanism, we propose another way for output feature aggregations. Instead of simple concatenation, we employ *averaging aggregation*, and delay applying final non-linearity function on each attention mechanism until averaging is performed:

$$\mathbf{v}'_{q_i} = \sigma\Big(\frac{1}{\mathcal{B}} \sum_{b=1}^{\mathcal{B}} \sum_{q_j \in \mathcal{N}_{q_i}} \alpha^b_{i,j} \mathbf{v}_{q_j}\Big). \tag{5}$$

The number of heads and the selection on these two multi-head attention aggregation methods are hyperparameters in GraphCM, and should be fine-tuned according to different tasks. After the GAT process, we encode the sequence of query embeddings $\{\mathbf{v}'_{q_1}, ..., \mathbf{v}'_{q_n}\}$ through a standard Gated Recurrent Unit (GRU) to generate the query context representations:

$$\mathbf{h}^q_i = \text{GRU}_{\text{query}}\left(\mathbf{v}'_{q_1}, \mathbf{v}'_{q_2}, ..., \mathbf{v}'_{q_i}\right),\ i = 1, ..., n. \tag{6}$$

4.4.2 Document Encoder. The document homogeneous graph mentioned in Section 4.2, on the other hand, preserves collaborative and similarity information among documents. Similar to the query encoder, to extract useful clues for user behaviors from documents, we first perform neighbor aggregation via GAT, and then apply GRU to generate document context representations.

The document encoder takes document ID d, vertical type v, previous click variable c and ranked position p as inputs and generates document context representations. Likewise, we apply the document homogeneous GAT on the document vectors after embedding layers. Neighbor sampling is also performed to form the neighbor sets $\mathcal{N}_{d_{i,j}}$ for $d_{i,j}$, which denotes the j-th document in the i-th query in the current session.

$$\begin{aligned} \beta_{i,j,k} &= \text{Softmax}_k(\text{att}_{\text{doc}}(\mathbf{v}_{d_{i,j}}, \mathbf{v}_{d_k})),\ d_k \in \mathcal{N}_{d_{i,j}}, \\ \mathbf{v}'_{d_{i,j}} &= \sum\nolimits_{d_k \in \mathcal{N}_{d_{i,j}}} \beta_{i,j,k} \mathbf{v}_{d_k}, \end{aligned} \tag{7}$$

where att_{doc} is an asymmetric node-level linear attention and is shared for all the nodes in the document homogeneous graph. We also apply a multi-head attention mechanism with two different feature aggregation methods to stabilize the learning process of the graph attention network:

$$\begin{aligned} \mathbf{v}'_{d_{i,j}} &= \Big\Vert_{b=1}^{\mathcal{B}} \sigma\Big(\sum\nolimits_{d_k \in \mathcal{N}_{d_{i,j}}} \beta^b_{i,j,k} \mathbf{v}_{d_k}\Big),\ or \\ \mathbf{v}'_{d_{i,j}} &= \sigma\Big(\frac{1}{\mathcal{B}} \sum_{b=1}^{\mathcal{B}} \sum_{d_k \in \mathcal{N}_{d_{i,j}}} \beta^b_{i,j,k} \mathbf{v}_{d_k}\Big). \end{aligned} \tag{8}$$

After obtaining the document representations from GAT, we concatenate all input embeddings and generate document context representations through a GRU:

$$\begin{aligned} \mathbf{x}_{i,j} &= [\mathbf{v}'_{d_{i,j}} \oplus \mathbf{v}_{v_{i,j}} \oplus \mathbf{v}_{c_{i,j}} \oplus \mathbf{v}_{p_{i,j}}], \\ \mathbf{h}^d_{i,j} &= \text{GRU}_{\text{doc}}\left(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, ..., \mathbf{x}_{i,j}\right). \end{aligned} \tag{9}$$

Here $\oplus$ denotes the vector concatenation operation, which shares the same meaning with the symbol $\|$.

4.4.3 Neighbor Interaction Module. It is essential to take interactions between queries and documents into account in the field of click models. However, the previous two components (i.e., query encoder and document encoder) only model the context information of queries and documents separately. We should further consider interactions between queries and documents. Moreover, instead of only considering the interaction between the current query q_i and document $d_{i,j}$, we propose a neighbor interaction method to explicitly incorporate the document's high-order neighbor information, which further enrich the local graph structural information and alleviate the data sparsity problem. Neighbors of the query are not considered since the query homogeneous graph is more sparse than the document homogeneous graph. Therefore, for the current query q_i and document $d_{i,j}$, we first apply neighbor sampling on homogeneous GATs to generate the document's neighbor set $\mathcal{N}_{d_{i,j}}$ and adjusted embeddings $\mathbf{v}'_q, \mathbf{v}'_d$, then we perform an attention-based feature interaction layer between the current query q_i and neighbor documents d_k in the neighbor set $\mathcal{N}_{d_{i,j}}$:

$$\begin{aligned} \mathbf{x}_{i,k} &= \mathbf{v}'_{q_i} \odot \mathbf{v}'_{d_k},\ d_k \in \mathcal{N}_{d_{i,j}}, \\ \gamma_k &= \text{Softmax}_k(\text{att}_{\text{inter}}(\mathbf{x}_{i,k})),\ d_k \in \mathcal{N}_{d_{i,j}}, \\ \mathbf{h}^i_{i,j} &= \sum\nolimits_{d_k \in \mathcal{N}_{d_{i,j}}} \gamma_k \mathbf{x}_{i,k},\ d_k \in \mathcal{N}_{d_{i,j}}, \end{aligned} \tag{10}$$

where $\odot$ denotes an element-wise product for vectors, and $\text{att}_{\text{inter}}$ is a shared asymmetric linear attention layer with a LeakyReLU activation function. It is worth noting that we do not perform the multi-head attention mechanism in the neighbor interaction module. Previous works [21] and our empirical experiments have shown that the multi-head attention mechanism is not much of a help for improving the performance, while contributing to the computational cost.

After generating the query context representation $\mathbf{h}^q_i$, the document context representation $\mathbf{h}^d_{i,j}$ and the neighbor interaction representation $\mathbf{h}^i_{i,j}$, we concatenate them together and put them through a two-layer Multi-Layer Perceptron (MLP) to output the estimated attractiveness score $\mathcal{A}_{i,j}$ for the current query q_i and document $d_{i,j}$:

$$\begin{aligned} \mathbf{x}_{i,j} &= [\mathbf{h}^q_i \oplus \mathbf{h}^d_{i,j} \oplus \mathbf{h}^i_{i,j}], \\ \mathcal{A}_{i,j} &= \text{MLP}(\mathbf{x}_{i,j}). \end{aligned} \tag{11}$$

Here, the activation functions for the first and the second layer in MLP are both LeakyReLU functions.

4.5 Examination Predictor

While the attractiveness estimator measures the attractiveness scores of each document to the user, the examination predictor aims to predict whether the user will continue to examine the document $d_{i,j}$ based on her session context. Following [8], we assume that a user's examination action is only affected by her actions on previous documents — the examination probability is not affected by the content of the document (users read the content only when the examination behavior happens). Therefore, for the current document $d_{n,m}$, we apply a session-level GRU to encode the

information of ranked position p, vertical type v and previous click variables c within the same session. Finally, a linear layer followed by a Sigmoid function is performed on the encoded hidden states to output the examination probability $\mathcal{E}$:

$$\begin{aligned} \mathbf{x}_{i,j} &= [\mathbf{v}_{p_{i,j}} \oplus \mathbf{v}_{v_{i,j}} \oplus \mathbf{v}_{c_{i,j}}], \\ \mathbf{h}^e_{i,j} &= \text{GRU}_{exam}\left(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, ..., \mathbf{x}_{i,j}\right), \\ \mathcal{E}_{i,j} &= \text{Sigmoid}\left(\text{Linear}\left(\mathbf{h}^e_{i,j}\right)\right). \end{aligned} \tag{12}$$

4.6 Click Predictor

The click predictor module combines the examination probability $\mathcal{E}$ and attractiveness score $\mathcal{A}$, and outputs the predicted click probability. We implement four different combination functions, which are shown in Table 1.

Table 1: Combination functions. E.H. is short for examination hypothesis. α and β are learnable parameters.

Function	Formula	Support E.H.?
mul	$c = \mathcal{E} \times \mathcal{A}$	Yes
expmul	$c = \mathcal{E}^{\alpha} \times \mathcal{A}^{\beta}$	Yes
linear	$c = \alpha\mathcal{E} + \beta\mathcal{A}$	No
nonlinear	$c = \text{MLP}(\mathcal{E}, \mathcal{A})$	No

The *mul* function simply follows the examination hypothesis and directly multiply the examination probability $\mathcal{E}$ with attractiveness score $\mathcal{A}$. The *expmul* function preserves the examination hypothesis and increases the model capacity by adding learnable parameters. The *linear* and *nonlinear* functions further explore the relation between $\mathcal{E}$ and $\mathcal{A}$ beyond the examination hypothesis.

After click prediction, we adopt a binary cross-entropy loss to ensure an end-to-end training of GraphCM. The objective function to be minimized during training is:

$$\mathcal{L}(\theta) = \mathcal{L}_c(\theta) + \lambda\|\theta\|^2 \tag{13}$$

$$\mathcal{L}_c(\theta) = -\frac{1}{N}\sum_{i,j} C_{i,j}\log\mathcal{P}_{i,j} + (1 - C_{i,j})\log(1 - \mathcal{P}_{i,j}) \tag{14}$$

where θ denotes trainable parameters in GraphCM, λ denotes the hyperparameter for regularization, N denotes the number of training batches, $C_{i,r}$ and $\mathcal{P}_{i,j}$ denote the real click signal and the predicted click probability.

5 EXPERIMENT

In this section, we conduct extensive experiments to answer the following questions:

- **RQ1** Does GraphCM mitigate the data sparsity problem and achieve the best performance compared with baseline models?
- **RQ2** Can GraphCM tackle the cold-start problem? How does GraphCM perform if it meets a brand-new query or document during the test phase?
- **RQ3** Which combination function performs best in integrating the attractiveness scores and examination probabilities?
- **RQ4** What is the influence of different components in GraphCM?

5.1 Experimental Setup

5.1.1 Dataset. We choose three real-world public session datasets in web search collected by different search engine platforms: Yandex[5], TREC2014[6] and TianGong-ST[7] [7]. The statistics of the datasets can be found in Table 2. Due to the memory limitation, we downsample the size of Yandex dataset. All datasets are divided into training, validation, and test sets with proportion 8:1:1. Besides, we would like to make two statements about the datasets before we elaborate on the details of the experiments:

(1) TianGong-ST is the only dataset that provides the vertical type information for each document in the search logs. For other datasets, we simply assume that all documents are presented in the same vertical type, and assign the same vector embedding for all vertical types.
(2) TianGong-ST is the only dataset that provides relevance labels for query-document pairs. Therefore, we perform the relevance estimation task only on the TianGong-ST dataset, and perform the click prediction task on all the three datasets.

Table 2: The dataset statistics

Dataset	Session	Query	Search Engine
Yandex	200,000	376,965	Yandex
TREC2014	1,257	5,443	Indri
TianGong-ST	147,155	356,252	Sogou

5.1.2 Baselines. The existing click models can be categorized into two classes: PGM-based and NN-based methods. In this paper, We consider CCM [14], DCM [15], DBN [5], SDBN [9], PBM [11] and UBM [12] as representative PGM-based click models, of which open-source implementations are available[8]. For NN-based click models, we consider NCM [3] and CACM [8] as baselines.

5.1.3 Evaluation Metrics. We compare GraphCM with baseline models based on the following two tasks: *click prediction* and *relevance estimation*, which are both common tasks for click models [9]. For click prediction task, we report the log-likelihood (LL) and perplexity (PPL) [3] of each model. The definitions of the log-likelihood and click perplexity at the rank r are as follows:

$$LL = \frac{1}{MN}\sum_{i=1}^{N}\sum_{r=1}^{M} C_{i,r}\log\mathcal{P}_{i,r} + (1 - C_{i,j})\log(1 - \mathcal{P}_{i,r}) \tag{15}$$

$$PPL@r = 2^{-\frac{1}{N}\sum_{i=1}^{N} c_{i,r}\log\mathcal{P}_{i,r} + (1-C_{i,r})\log(1-\mathcal{P}_{i,r})} \tag{16}$$

where subscript r is the ranked position, N is the total number of queries, and M is the number of documents in each query. $C_{i,r}$ and $\mathcal{P}_{i,r}$ are the real click signal and the predicted click probability of the r-th document in the i-th query. We calculate the total perplexity by averaging perplexities over all the positions. Lower values of perplexity and higher values of log-likelihood correspond to better click prediction performance.

For the relevance estimation task, we use click models to rank the document list and compute the averaged Normalized Discounted

[5]https://www.kaggle.com/c/yandex-personalized-web-search-challenge
[6]https://trec.nist.gov/data/session2014.html
[7]http://www.thuir.cn/tiangong-st/
[8]https://github.com/markovi/PyClick

Table 3: Overall performance of each click model. We only perform relevance estimation task on TianGong-ST, since it is the only dataset that provides human-annotated relevance labels. The best results are given in bold, while the second best values are underlined. $*$ indicates statistically significant improvement (measured by t-test) with p-value < 0.001 over all baselines. *Note*: LL, the higher, the better; PPL, the lower, the better; NDCG, the higher, the better.

Model	Yandex		TREC2014		TianGong-ST					
	LL	PPL	LL	PPL	LL	PPL	NDCG@1	NDCG@3	NDCG@5	NDCG@10
CCM	-0.4171	1.3436	-0.5445	1.3267	-0.2057	1.2238	0.6596	0.6931	0.7175	0.8453
DCM	-0.4183	1.3226	-0.5485	1.2998	-0.2109	1.2231	0.6861	0.6818	0.7115	0.8441
DBN	-0.4157	1.3397	-0.5451	1.3216	-0.2052	1.2292	0.6883	0.6997	0.7261	0.8501
SDBN	-0.4174	1.3399	-0.5491	1.3218	-0.2094	1.2293	0.6814	0.6777	0.7104	0.8421
PBM	-0.2429	1.2961	-0.1678	1.1899	-0.1825	1.2183	0.6594	0.6422	0.6703	0.8243
UBM	-0.2275	1.2662	-0.1568	1.1901	-0.1751	1.2179	0.6362	0.6354	0.6671	0.8208
NCM	-0.2204	1.2666	-0.1549	1.1757	-0.1718	1.2055	0.7081	0.7066	0.7368	0.8621
CACM	-0.2199	1.2662	-0.1541	1.1738	-0.1707	1.2027	0.7347	0.7163	0.7405	0.8667
GraphCM	**-0.2192***	**1.2652***	**-0.1529***	**1.1721***	**-0.1629***	**1.1938***	**0.7388***	**0.7189***	**0.7466***	**0.8671***

Table 4: Data sparsity statistics and LL/PPL improvements. The sparsity is calculated as # missing query-document interactions / # possible query-document pairs. The improvements are absolute LL/PPL gains of GraphCM compared with the best baseline.

Dataset	Sparsity	LL Imprv.	PPL Imprv.
Yandex	99.8895%	0.32%	0.08%
TREC2014	99.9189%	0.78%	0.14%
TianGong-ST	99.9969%	4.57%	0.74%

Cumulative Gain (NDCG) [17] according to the human-annotated relevance labels. We report NDCG scores at truncation level 1, 3, 5, and 10. Higher values of NDCG indicates better relevance estimation performance.

5.1.4 Implementation Details. We train GraphCM with a mini-batch size of 128 with Adam optimizer. The hidden sizes of all GRUs are 64. The embedding sizes of query q, document d, vertical type v, click variable c and ranked position p are 64, 64, 8, 4, 4 respectively. The initial learning rate is selected from $\{10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$. To avoid overfitting, we choose the coefficient of L2 norm and dropout rate from $\{10^{-4}, 10^{-5}\}$ and $\{0.25, 0.5\}$. The number of neighbors to be sampled is tuned from $\{1, 2, 4, 8, 16, 32\}$. Finally, we adopt the model at the iteration with the lowest validation PPL for evaluation in the test set. To ensure fair comparison, we also fine-tune all the baseline models to achieve their best performance. Our model implemented on PyTorch is available [9].

5.2 Performance Comparison (RQ1)

We perform click prediction and relevance estimation tasks on each click model for performance comparison. It is worth noting that the relevance estimation task is only performed on TianGong-ST dataset, as it is the only dataset that provides human-annotated relevance labels. The results are presented in Table 3, from which we can obtain the following observations:

(1) All NN-based models show significant improvements over PGM-based models in click prediction and relevance estimation tasks. NN-based models adopt the distributed vector representation approach for representations of queries and documents, therefore can better capture user behavior patterns.
(2) CACM achieves the best performance among all the baseline models, followed by NCM and other PGM-based click models. CACM utilizes complex model structures to take the intra-session information into consideration and thus better capture the user behavior patterns, which is consistent with the results reported in [8].
(3) GraphCM significantly outperforms all the baseline models. Such improvement validates the effectiveness of applying graph neural networks and neighbor interaction techniques to make use of the inter-session and intra-session information, which enables GraphCM to capture more subtle patterns in user click behaviors.

Following [21], to study the ability of GraphCM to tackle the data sparsity problem, we further investigate the data sparsity statistics and improvements of GraphCM compared to the best baseline model among different datasets. The results are shown in Table 4. As the data sparsity ratios increase (Yandex: 99.8895%; TREC2014: 99.9189%; TianGong-ST: 99.9969%;), the GraphCM gradually achieves better improvements for LL and PPL metrics. This means that GraphCM exploits intra-session and inter-session information from the pre-constructed homogeneous graphs via graph neural networks and neighbor interaction techniques, thus leading to better performance in mitigating the data sparsity problem.

5.3 Cold-start Problem (RQ2)

Similar to works in recommender systems, existing click models suffer from the cold-start problem. Namely, there are brand-new queries or documents during the test phase that never appear in the training set. To ensure the unambiguity, before we zoom into the details of experiments, we first define the following concepts. Suppose we have a session $\mathcal{S}$ in the test set, which consists of a sequence of queries $\mathcal{Q} = \{q_1, ..., q_n\}$ and documents $\mathcal{D} = \{d_{1,1}, ..., d_{n,m}\}$, we propose the following definitions:

[9] https://github.com/CHIANGEL/GraphCM

Table 5: LL (the higher, the better) and PPL (the lower, the better) performance of click models for the cold-start problems on different dataset. The best results are given in bold, while the second best values are underlined. $*$ indicates statistically significant improvement (measured by t-test) with p-value < 0.001 over all baselines.

Metric	Model	Yandex				TREC2014				TianGong-ST			
		Cold Q	Cold D	Cold QD	Warm QD	Cold Q	Cold D	Cold QD	Warm QD	Cold Q	Cold D	Cold QD	Warm QD
LL	UBM	-0.2418	-0.2397	-0.2346	-0.1755	-0.3105	-0.2021	-0.1408	-0.1432	-0.2157	-0.2011	-0.2189	-0.1696
	NCM	-0.2386	-0.1841	-0.2343	-0.1685	-0.3069	-0.1717	-0.1398	-0.1422	-0.1987	-0.2009	-0.2091	-0.1609
	CACM	-0.2319	-0.1798	-0.2338	-0.1626	-0.3045	-0.1711	-0.1389	-0.1421	-0.1977	-0.1979	-0.2076	-0.1586
	GraphCM	**-0.2316***	**-0.1792***	**-0.2324***	**-0.1624***	**-0.2954***	**-0.1645***	**-0.1379***	**-0.1418***	**-0.1948***	**-0.1977***	**-0.2074***	**-0.1527***
PPL	UBM	1.3265	1.3174	1.3269	1.2242	1.4008	1.2234	1.1608	1.1688	1.2473	1.2689	1.2679	1.2082
	NCM	1.2931	1.2254	1.2847	1.2044	1.3967	1.2153	1.1597	1.1679	1.2363	1.2441	1.2539	1.1919
	CACM	1.2829	1.2185	1.2839	1.1957	1.3564	1.2106	1.1581	1.1677	1.2355	1.2392	1.2513	1.1885
	GraphCM	**1.2823***	**1.2176***	**1.2823***	**1.1956***	**1.2381***	**1.2013***	**1.1578***	**1.1676***	**1.2336***	**1.2389***	**1.2508***	**1.1792***

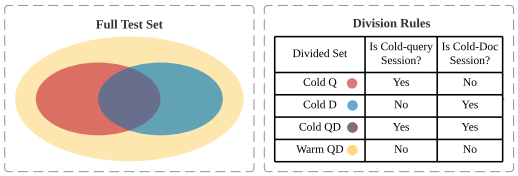

Figure 5: The full test set division for cold-start problems.

Definition 5.1. Define a query q in the test set as a **cold query** if and only if it never appears in the training set.

Definition 5.2. Define a query q in the test set as a **warm query** if and only if it has appeared in the training set.

Definition 5.3. Define the session $\mathcal{S}$ as a **cold-query session** if and only if there exists at least one *cold query* $q_i \in \mathcal{Q}$.

Definition 5.4. Define the session $\mathcal{S}$ as a **warm-query session** if and only if all the queries $q_i \in \mathcal{Q}$ are *warm queries*.

Note that all the definitions above focus on the *query cold-start* problem (i.e., new queries in the test set). We can further propose the parallel concepts for the *document cold-start* problem, which is omitted here due to page limitations. Based on the definitions above, as illustrated in Figure 5, we can split the full test set into four mutually exclusive session sets: (i) Cold Q set; (ii) Cold D set; (iii) Cold QD set; (iv) Warm QD set. The four session sets have varying degrees on the cold-start problem. Then we measure the LL and PPL performance of GraphCM, CACM, NCM and UBM on these session sets respectively for different datasets (NDCG metric is omitted since it shows similar trends). The results are shown in Table 5, from which we can obtain the following observations:

(1) Generally, click models achieve much worse performance on the Cold Q set, Cold D set and Cold QD set compared to the performance on the Warm QD set, which validates the fact that click models are quite vulnerable to the cold-start problems. However, for TREC2014 dataset, we observe that the performance of click models on Cold QD set is better than it on Warm QD set. A possible reason is that Cold QD session dominates the validation set of TREC2014 dataset (up to 62.91%). The model selection on such a validation set let the model fit the Cold QD data better.
(2) As a PGM-based model, UBM achieves the worst performance compared to NN-based methods on the four session sets, which suggests that the distributed vector representation approach adopted by NN-based methods is better than the traditional binary random variable representations in cold-start problems.
(3) CACM achieves relatively better performance than NCM on the four session sets for three datasets. As CACM applies GRU units to encode the intra-session information during the test phase, it can somehow implicitly utilize the reformulation information between queries and similarity information between documents, which helps cope with the cold-start problems.
(4) GraphCM outperforms all baselines on four session sets for three datasets. Different from CACM's implicit utilization of the reformulation and similarity information, GraphCM directly model these relations in the homogeneous graphs, and applies graph neural networks and neighbor interaction techniques to explicitly leverage the auxiliary information as guidance for user behavior modeling and click prediction. The results demonstrate that GraphCM can better tackle the cold start problems.
(5) The full test set, as a union of four session sets, is more similar to real-world application scenarios where cold queries or documents are likely to show up among the warm ones. As presented in Table 3, GraphCM achieves the best performance on the full test set, which implies the effectiveness of GraphCM to tackle the cold-start problem in real-world applications.

5.4 Combination Function (RQ3)

We study the effectiveness of different combination functions by comparing their click prediction performance, which is shown in Figure 6. From Figure 6, we can obtain the following observations:

(1) GraphCMs with the *mul* and *expmul* functions outperform those with the *linear* and *nonlinear* functions. This indicates that combination functions that support the examination hypothesis can properly represent user behavior patterns and achieve better performance.
(2) GraphCM with *expmul* function achieves the best performance among all combination functions. Although the *expmul* and *mul* functions both support the examination hypothesis, the *mul* function is only a special case of the *expmul* function (i.e., $\alpha = 1, \beta = 1$). Therefore, the *expmul* function can model user behaviors more flexibly with more learnable parameters.
(3) In theory, the *nonlinear* function, as a two-layer fully connected network, is able to cover the computational formulas of other combination functions and perform at least as good as other functions. However, the performance of *nonlinear*

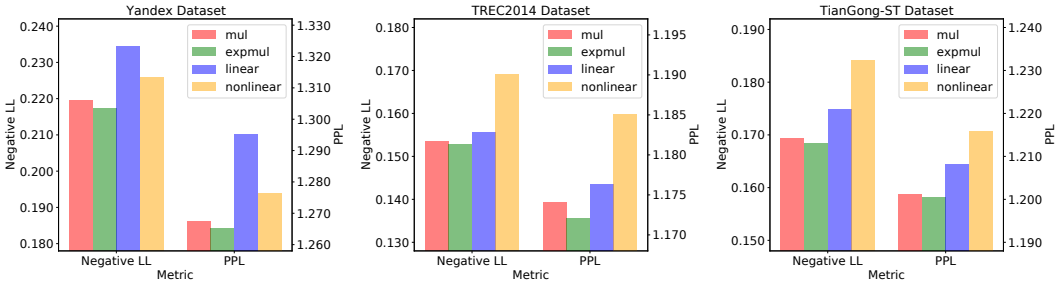

Figure 6: GraphCM's click prediction performance with different combination functions. *Note*: Negative LL, the lower, the better; PPL, the lower, the better.

Table 6: The learnable parameters α and β for combination functions *expmul* and *linear*.

Function	Formula	Yandex		TREC2014		TianGong-ST	
		α	β	α	β	α	β
expmul	$c = \mathcal{E}^{\alpha} \times \mathcal{A}^{\beta}$	1.035	0.991	1.031	1.010	0.952	0.796
linear	$c = \alpha\mathcal{E} + \beta\mathcal{A}$	0.468	0.462	0.463	0.437	0.491	0.401

function shown in Figure 6 is much worse than other functions. We argue that the *nonlinear* function overfits on the click signals and fails to learn the multiplication transformation that supports the examination hypothesis, resulting in poor performance.

In function *expmul* and *linear*, the parameters α and β are learnable, instead of hyperparameters that require manual assignments. To further investigate their learning mechanism, we check the values of the learnable parameters in functions *expmul* and *linear*. The results are shown in Table 6. We observe that α is always higher than β for different combination functions and datasets. This indicates that GraphCM assigns higher weights to the attractiveness scores generated by the attractiveness estimator, since the structure of attractiveness estimator is more complex and is able to encode more important information for user behavior modeling.

5.5 Ablation Study (RQ4)

In order to investigate the contribution of each component to the final performance of GraphCM, we conduct several comparison experiments by removing the proposed query homogeneous GAT, document homogeneous GAT, and neighbor interaction module respectively. The results are presented in Table 7 and 8. When we remove the query or document homogeneous GAT, the performance degrades on three dataset for both two tasks, which suggests that graph neural networks can extract the collaborative, reformulation and similarity information in the pre-constructed homogeneous graphs to help user behavior prediction. Likewise, the performance drops for both click prediction and relevance estimation tasks on three datasets when we remove the neighbor interaction module, which validates the effectiveness of applying neighbor interactions to capture local and global structural information [21].

6 CONCLUSION

In this work, we propose a novel graph-enhanced click model (GraphCM) for web search. We separately model the attractiveness estimation and examination predictor, and apply graph neural networks and neighbor interaction techniques to exploit intra-session and inter-session information for user behavior prediction. Extensive experiments are conducted on three real-world session datasets, which validates the effectiveness of our solution in addressing the data sparsity and cold-start problems.

Table 7: The comparison of LL and PPL w.r.t. homogeneous GATs and neighbor interaction module. We perform the following operations respectively: i. remove the query homogeneous GAT; ii. remove the document homogeneous GAT; iii. remove the neighbor interaction module. The best results are given in bold. $*$ indicates statistically significant improvement (measured by t-test) with p-value < 0.001.

Model	Yandex		TREC2014		TianGong-ST	
	LL	PPL	LL	PPL	LL	PPL
0. GraphCM	**-0.2192***	**1.2652***	**-0.1529***	**1.1721***	**-0.1629***	**1.1938***
i. w/o Q.GAT	-0.2201	1.2667	-0.1601	1.1818	-0.1656	1.1968
ii. w/o D.GAT	-0.2198	1.2672	-0.1532	1.1738	-0.1637	1.1943
iii. w/o Neigh.	-0.2304	1.2755	-0.1608	1.1801	-0.1702	1.2031

Table 8: The comparison of NDCG performance w.r.t. homogeneous GATs and neighbor interaction module for TianGong-ST dataset. The performed operations stay the same as Table 7. The best results are given in bold. $*$ indicates statistically significant improvement (measured by t-test) with p-value < 0.001.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10
0. GraphCM	**0.7388***	**0.7189***	**0.7466***	**0.8671***
i. w/o Q.GAT	0.7197	0.7092	0.7351	0.8632
ii. w/o D.GAT	0.6985	0.7045	0.7323	0.8593
iii.w/o Neigh.	0.7209	0.7104	0.7382	0.8642

ACKNOWLEDGEMENT

The corresponding author Weinan Zhang is supported by "New Generation of AI 2030" Major Project (2018AAA0100900) and National Natural Science Foundation of China (62076161, 61772333, 61632017). The work is also sponsored by Huawei Innovation Research Program. We thank Student Innovation Center at Shanghai Jiao Tong University for the provision of GPU computing resources. We thank MindSpore [1] for the partial support of this work, which is a new deep learning computing framework.

REFERENCES

[1] 2020. MindSpore. https://www.mindspore.cn/

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

[3] Alexey Borisov, Ilya Markov, Maarten De Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*. 531–541.

[4] Alexey Borisov, Martijn Wardenaar, Ilya Markov, and Maarten de Rijke. 2018. A click sequence model for web search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 45–54.

[5] Olivier Chapelle and Ya Zhang. 2009. A Dynamic Bayesian Network Click Model for Web Search Ranking. In *Proceedings of the 18th International Conference on World Wide Web*. 1–10.

[6] Jia Chen. 2020. Beyond Sessions: Exploiting Hybrid Contextual Information for Web Search. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 915–916.

[7] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. TianGong-ST: A New Dataset with Large-scale Refined Real-world Web Search Sessions. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2485–2488.

[8] Jia Chen, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. A Context-Aware Click Model for Web Search. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 88–96.

[9] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. Click models for web search. *Synthesis lectures on information concepts, retrieval, and services* 7, 3 (2015), 1–115.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[11] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*. 87–94.

[12] Georges E Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations.. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 331–338.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[14] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. 2009. Click Chain Model in Web Search. In *Proceedings of the 18th International Conference on World Wide Web*. 11–20.

[15] Fan Guo, Chao Liu, and Yi Min Wang. 2009. Efficient Multiple-Click Models in Web Search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*. 124–131.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[17] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR Evaluation Methods for Retrieving Highly Relevant Documents. *SIGIR Forum* (2017), 243–250.

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

[19] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[21] 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.

[22] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 385–394.

[23] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. 521–530.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of International Conference on Learning Representations*.

[26] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.

[27] Hongning Wang, ChengXiang Zhai, Anlei Dong, and Yi Chang. 2013. Content-aware click modeling. In *Proceedings of the 22nd international conference on World Wide Web*. 1365–1376.

[28] 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.

[29] Danqing Xu, Yiqun Liu, Min Zhang, Shaoping Ma, and Liyun Ru. 2012. Incorporating revisiting behaviors into click models. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 303–312.

[30] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[31] 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.

[32] Yuchen Zhang, Weizhu Chen, Dong Wang, and Qiang Yang. 2011. User-click modeling for understanding and predicting search-behavior. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1388–1396.

[33] Feimin Zhong, Dong Wang, Gang Wang, Weizhu Chen, Yuchen Zhang, Zheng Chen, and Haixun Wang. 2010. Incorporating post-click behaviors into a click model. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 355–362.

[34] Zeyuan Allen Zhu, Weizhu Chen, Tom Minka, Chenguang Zhu, and Zheng Chen. 2010. A novel click model and its applications to online advertising. In *Proceedings of the third ACM international conference on Web search and data mining*. 321–330.