Joint Learning of E-commerce Search and Recommendation with a Unified Graph Neural Network

Kai Zhao*‡, Yukun Zheng*, Tao Zhuang†, Xiang Li‡, and Xiaoyi Zeng
Alibaba Group, Beijing, China
{weimu.zk,zyk265182,zhuangtao.zt,leo.lx}@alibaba-inc.com,yuanhan@taobao.com

ABSTRACT

Click-through rate (CTR) prediction plays an important role in search and recommendation, which are the two most prominent scenarios in e-commerce. A number of models have been proposed to predict CTR by mining user behaviors, especially users' interactions with items. But the sparseness of user behaviors is an obstacle to the improvement of CTR prediction. Previous works only focused on one scenario, either search or recommendation. However, on a practical e-commerce platform, search and recommendation share the same set of users and items, which means joint learning of both scenarios may alleviate the sparseness of user behaviors. In this paper, we propose a novel **S**earch and **R**ecommendation **J**oint **G**raph (**SRJGraph**) neural network to jointly learn a better CTR model for both scenarios. A key question of joint learning is how to effectively share information across search and recommendation, in spite of their differences. A notable difference between search and recommendation is that there are explicit queries in search, whereas no query exists in recommendation. We address this difference by constructing a unified graph to share representations of users and items across search and recommendation, as well as represent user-item interactions uniformly. In this graph, users and items are heterogeneous nodes, and search queries are incorporated into the user-item interaction edges as attributes. For recommendation where no query exists, a special attribute is attached on user-item interaction edges. We further propose an intention and upstream-aware aggregator to explore useful information from high-order connections among users and items. We conduct extensive experiments on a large-scale dataset collected from *Taobao.com*, the largest e-commerce platform in China. Empirical results show that SRJGraph significantly outperforms the state-of-the-art approaches of CTR prediction in both search and recommendation tasks.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; **Recommender systems**; **Personalization**; **Online shopping**.

KEYWORDS

Click-through rate prediction, graph neural network, joint learning, product search and recommendation, e-commerce

ACM Reference Format:
Kai Zhao, Yukun Zheng, Tao Zhuang, Xiang Li, and Xiaoyi Zeng. 2022. Joint Learning of E-commerce Search and Recommendation with a Unified Graph Neural Network. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498414

WSDM '22, February 21–25, 2022, Tempe, AZ, USA.

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498414

1 INTRODUCTION

In e-commerce platforms, accurately estimating click-through rate (CTR) has an essential impact on the revenue of online retailers as well as the experience of customers. Therefore, the CTR prediction problem has attracted extensive attention in the communities of both academia and industry [21]. Recently, many deep learning-based models [19, 20, 39, 40] have been proposed to add this problem. Benefiting from deep learning, they can exploit non-linear relations among plenty of features. Among these features, user behaviors, especially the items clicked by the user, are found extremely helpful for CTR prediction [39, 40]. However, on a real e-commerce platform, clicks are usually sparse from the views of both users and items. From the user view, many users are inactive and only click a few items in a week. From the item view, there are billions of items on large e-commerce platforms such as Taobao, but most of them attract few clicks from users. Therefore, the sparseness problem of user behaviors hinders the improvement of CTR prediction task.

As we know, search and recommendation are usually the two most prominent scenarios visited by users in an e-commerce platform. It is reasonable to perform joint learning of search and recommendation models by utilizing data from both scenarios to alleviate the behavior sparseness problem. Taking a real-world dataset (shown in Section 5.1) for example, each user has 226 requests and 113 page-views in search and recommendation engine on average, but only has 93 and 61 click behaviors, separately. Among all clicked items of one user, only 2% of them were clicked by the user in both search and recommendation on average. Thus, the abundant user behaviors on one scene may alleviate the behavior sparseness problem in the other scene.

However, few previous works have studied joint learning of CTR prediction model for e-commerce search and recommendation. In

* Both authors contributed equally to this research.
† Corresponding author.
‡ Participated in this work while at Alibaba.

this paper, we propose a novel CTR prediction model to perform this kind of joint learning. Here we list the two challenges of joint learning for search and recommendation and accordingly introduce how our model addresses these challenges:

The first challenge is how to effectively share information between search and recommendation in the face of obvious differences between them. The biggest difference between search and recommendation is that there are explicit queries in search, whereas no query exists in recommendation.

With the power of Graph Neural Networks (GNN) [7, 10, 17], we tackle this challenge by proposing a **S**earch and **R**ecommendation **J**oint **G**raph (**SRJGraph**) neural network. To be more specific, we first construct a heterogeneous attributed graph [31] to represent user-item interactions from both search and recommendation, where users and items are the heterogeneous nodes and there is an edge between a user node and an item node if the user once clicked the item. Search queries are incorporated into the user-item interaction edges as attributes. As for recommendation where no query exists, a special query attribute is attached on the corresponding edges. Then we utilize SRJGraph to learn and aggregate information across search and recommendation data.

The second challenge is how to aggregate information from such a heterogeneous attributed graph to boost the performance of CTR prediction. Given the target user and item combination, the CTR prediction task is to predict whether the user will click this item when searching a certain query or browsing the recommendation list. By following traditional GNN models [10, 15, 26], recent GNN-based CTR prediction models [12, 29, 35] use the same aggregation operations, where the aggregation in each layer considers the local information of the central node and its neighbors. And this kind of aggregation generates the same representation for a user or an item, even though it might be different in different target combinations, which is not effective enough in the CTR prediction task.

To overcome this problem, we propose a novel aggregation layer that further leverages the information of the target user, query and item combination, and the precursor information on the aggregation path, rather than only taking the local information: 1) Our aggregation layer in SRJGraph distills useful information from the target user, query and item, which we call intention-aware aggregation in this paper. 2) In each step of the recursive aggregation process, we also take advantage of the information from the precursor node and the associated edge with the central node, which is called upstream-aware aggregation in this paper. Therefore, our aggregation layer has the capability to dynamically aggregate high-order user-item interaction information with respect to different targets.

Our contributions in this paper are summarized as follows:

- We are the first to jointly model CTR prediction of search and recommendation with a unified graph neural network.
- We propose a novel intention- and upstream-aware aggregation layer in our SRJGraph model to better aggregate high-order information for CTR prediction.
- By empirically evaluating on a large real-world dataset, we show that SRJGraph outperforms the state-of-the-art baselines in the CTR prediction task of both search and recommendation.

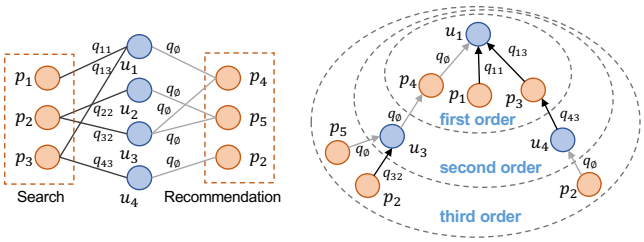
2 RELATED WORK

2.1 Search and Recommendation Models in E-commerce

2.1.1 Search Models. Search is the major tool for users to look for products they want to buy in e-commerce. In recent years, personalized product search started to get attention in both academia and industry. However, compared to personalized recommendation techniques, works on personalized product search are fewer. Two pioneering works [23, 24] analyzed the taxonomy of queries in product search and investigated search intents qualitatively. Ai et al. [2] analyzed the impact of personalization on a real e-commerce search engine and proposed an attention-based embedding model named AEM. Bi et al. [4] proposed a transformer-based embedding model named TEM, which uses the representations of the searching query and the items in the user behavior history as the inputs to the transformer. Zhang et al. [38] proposed a ranking model for product search named GEPS, which learns the embedding of queries and items using GNN.

2.1.2 Recommendation Models. Collaborative filtering (CF) assumes that users with similar behaviors have similar preferences on items. A number of modern recommender systems have been developed based on this assumption, such as Matrix Factorization (MF) [18]. Then the data-driven methods tried to leverage extra information such as item attributes [5], user reviews [6], social relations [28]. The other network-enhanced methods improved the user-item interaction function in MF by using nonlinear neural networks, such as NeuMF [14], NFM [11] and xDeepFM [19]. Recently, several works [12, 29, 40] utilized historical user-item interactions to better learn users' interests and preferences. Zhou et al. [40] proposed DIN to capture users' interests by exploiting users' historical behaviors. Moreover, a number of graph neural network-based methods have been proposed, such as BiRank [13], HOP-Rec [34], NGCF [29] and LightGCN [12]. Wang et al. [29] proposed NGCF to modeling users' preferences with high-order user-item interaction graph. He et al. [12] discarded the feature transformation and nonlinear activation of NGCF and proposed a more concise and effective model named LightGCN.

2.1.3 Joint Learning Models. In both academia and industry, only a few works studied joint learning on search and recommendation. Wang et al. [27] proposed a multinomial logistic regression model to integrate search features and recommendation features. Zamani and Croft [36] proposed a general joint framework to learn a search model and a recommendation model, named JSR. The search model in JSR learns the textual relevance between the search query and candidate items but does not consider user personalization which is very important for e-commerce. They also proposed another JSR model [37], where the search model aims to reconstruct items' textual descriptions, rather than predict the CTR. So we compare with the first JSR model in our experiments. Different from the two JSR models, our SRJGraph model aims to predict CTR for both personalized product search and recommendation in e-commerce. In addition, we utilize a joint GNN to effectively share useful information between search and recommendation.



(a) User-item interaction graph in search and recommendation.

(b) High-order connectivity for user u_1 in the graph of search and recommendation.

Figure 1: An illustration of the graph we constructed for search and recommendation.

2.2 Graph Neural Network

Graph Neural Network (GNN) [22] and Graph Convolutional Network (GCN) [7, 17] were proposed to utilize neural networks to process data represented in graphs. Hamilton et al. [10] proposed a general inductive framework named GraphSAGE, which generates a node's embedding by sampling and aggregating the embeddings of its neighborhoods. Besides, Velickovic et al. [26] proposed GAT, which utilizes the shared attention parameters to weigh different importance of node's neighbors. Since then, many variants of graph neural networks have been proposed to model different kinds of graphs, such as directed graph [16], heterogeneous graph [30], edge-informative graph [3], and temporal graph [15].

3 SEARCH AND RECOMMENDATION GRAPH

In this section, we introduce how to construct a joint search and recommendation graph and formalize our CTR prediction problem. Our joint graph is constructed from users' clicks from both search and recommendation. Let $\mathcal{U} = \{u_1, \cdots, u_{|\mathcal{U}|}\}$ denotes the set of users, and let $\mathcal{P} = \{p_1, \cdots, p_{|\mathcal{P}|}\}$ denotes the set of products, where $|\mathcal{U}|$ and $|\mathcal{P}|$ are the numbers of distinct users and products, respectively. A user click c is represented as $\langle u, p, q, t\rangle$, which means that user u clicked item p at the time t, and q denotes the query. If this click is from search, then q is the corresponding query which can be segmented into several shorter terms: $q = (w_1, w_2, \cdots, w_{|q|})$, where w_i denotes the i-th term and $|q|$ is the number of terms in query q. If this click is from recommendation where no query term exists, q is set to an empty query which we denote as q_ϕ. The set of all user click behaviors is denoted as $\mathcal{B}$. Using these aforementioned notations, we define our graph as follows:

Definition 1. *The **search and recommendation joint graph** is defined as $G = (V, E)$, where $V = \mathcal{U} \cup \mathcal{P}$ is the set of user and product nodes, $E = \mathcal{B}$ is the set of edges. Each edge $e \in E$ is a quadruple $\langle u, p, q, t\rangle$, meaning that user u clicked item p with query q at the time t.*

Figure 1(a) shows an illustrative example of our graph. User u_1 searches query q_{11} in search engine and then clicks item p_1, so there exists an edge between node u_1 and p_1, and query q_{11} becomes an attribute of this edge. When user u_1 clicks item p_4 in recommender system, there exists an edge between them with an assigned empty query q_ϕ. We don't treat the query as the third node type in the unified graph for the following reason: When queries serve as the third type of node, a search click is presented in the graph by two edges, i.e., user-query and query-item. But it couldn't handle the case where two users search the same query and click different items. At this time, the user-item edge is also needed to tell which item was clicked by which user. In comparison, we consider that our graph is more concise and easy to implement.

Our problem of CTR prediction is to predict whether a user will click an item when it is presented to the user, either in the search engine or in the recommender system. Note that most items to be predicted in search are relevant to the queries. Formally, a data sample is represented as $(\langle u, q, p\rangle, y)$, where $\langle u, q, p\rangle$ contains all input features and $y \in \{0, 1\}$ is the label. $\langle u, q, p\rangle$ means item p is presented to user u under query q, where $q = q_\phi$ if this sample is from recommendation. Given the input $\langle u, q, p\rangle$, our model needs to predict its label y, *i.e.,* whether user u will click item p.

4 MODEL FRAMEWORK

In this section, we present our model, SRJGraph, with three components: embedding layer, aggregation layers, and prediction layer. At the end of this section, we discuss the differences between our model and existing GNN-based models.

4.1 Embedding Layer

We use separated embedding look-up tables to represent IDs of users, items and query terms in our sparse-encoded data. We also maintain additional embedding look-up tables for multiple types of attributes associated with users and items (*e.g.,* user gender and item category). Then the representation vectors of user u and item p are generated by concatenating the embeddings of their IDs and their attributes, respectively:

$$e_u = e^{ID_u} || e^{a_1} || \cdots || e^{a_N} \tag{1}$$

$$e_p = e^{ID_p} || e^{b_1} || \cdots || e^{b_M} \tag{2}$$

where e^{ID_u} and e^{ID_p} are the embeddings for the IDs of u and p, e^{a_i} and e^{b_j} are the embeddings for the i-th attribute of u and the j-th attribute of p, N and M are the numbers of distinct attributes of u and p, respectively As we consider each query q as a sequence of terms $(w_1, w_2, \cdots, w_{|q|})$, we represent query q as the combination of its terms' embeddings:

$$e_q = g(e_{w_1}, e_{w_2}, \cdots, e_{w_{|q|}}) \tag{3}$$

where $e_{w_{q_k}}$ is the embedding of the k-th query term w_k, $g(\cdot)$ denotes a combination function. In this study, we have empirically found that element-wise sum-pooling is both efficient and effective for query term combination. For the recommendation data, we represent the special query q_ϕ by padding it to a fixed length with a unique term ID that doesn't appear in any other queries.

4.2 Aggregation Layer

The aggregation layer consists of two stages: forward multi-order neighbor sampling and backward node representation aggregation. We first introduce necessary background knowledge on GNN models. Then we describe the aggregation layer of SRJGraph.

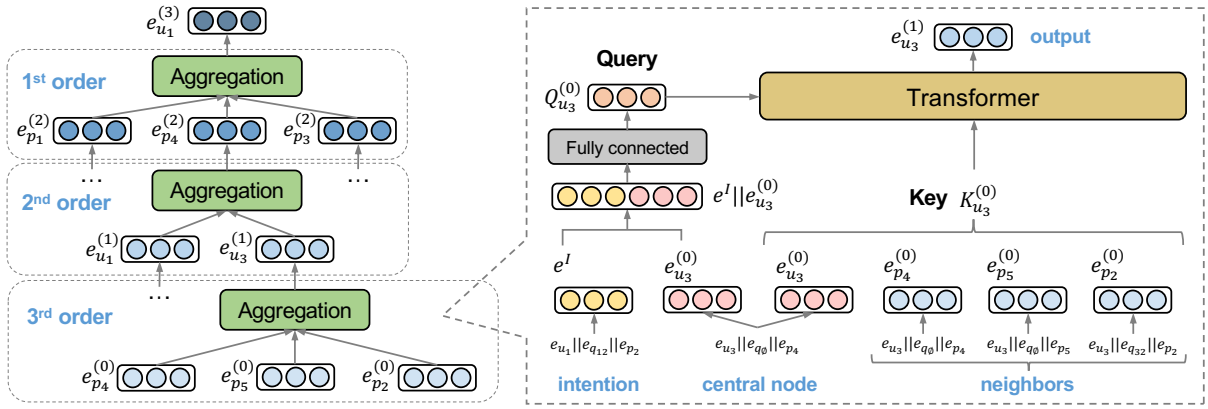


Figure 2: An illustration of the 3-rd order representation aggregation for user u_1 in Figure 1, assuming that the $\langle u_1, q_{1,2}, p_2 \rangle$ is the input sample to be predicted. After forward neighbor sampling up to 3-rd order, the representations are aggregated by transformer unit backward. The right part shows the details for the 1-st aggregation layer, *i.e.,* aggregation 3-rd order neighbors information to 2-nd order node u_3.

4.2.1 Background. GNN aims to learn the node's representation with the information of its neighbors on graph-structured data. Basically, a GNN layer can be defined as:

$$e_i^{(l)} = Aggregate\left(\left\{e_j^{(l-1)} \middle| j \in \mathcal{N}_i\right\} \cup \left\{e_i^{(l-1)}\right\}\right) \tag{4}$$

where $e_i^{(l)}$ is the refined representation of the node i in l-th aggregation layer, and $\mathcal{N}_i$ is a randomly sampled subset of the first-order neighbors of node i. The aim of random neighbor sampling is to reduce computation cost on large graphs [10].

4.2.2 Forward Multi-order Neighbor Sampling. Here we introduce how to sample neighbors in our SRJGraph model. With the sample $\langle u, q, p \rangle$ to be predicted, we will sample K-order neighbors for nodes u and p respectively, on our search and recommendation joint graph. For example, given the sample input $\mathbf{x} = \langle u, q, p \rangle$, user u's 1-st order neighbors are sampled as follows:

$$\mathcal{N}_u^1 = \left\{p' | p' \in \mathcal{P} \wedge \epsilon_{u,p'} \in E \wedge f(p', p)\right\} \tag{5}$$

where $\epsilon_{u,p'}$ denotes the edge between u and p', $f(\cdot)$ is a sampling function which decides whether to keep the neighbor node p' in $\mathcal{N}_u^1$. Note that $\mathcal{N}_u^1$ is not only determined by u, but also determined by the target item p due to the introduction of $f(p', p)$ in Equation 5. Filtering rules, such as attribute matching between neighbor node p' and target item p, can be incorporated to $f(p', p)$. We also record the direct precursor for each neighbor node. For each neighbor node $p' \in \mathcal{N}_u^1$, its **direct precursor** $pre(p')$ is u. Then the k-th order neighbors of node u are obtained recursively as:

$$\mathcal{N}_u^k = \bigcup_{n \in \mathcal{N}_u^{k-1}} \mathcal{N}_n^1 \tag{6}$$

Analogously, we can obtain the neighbors of the target item p in the input sample $\mathbf{x}$. Specifically, we adopt different sampling functions $f(\cdot)$ for user and item nodes. For a user node, we randomly sample a number of its neighbor item nodes which have the same category attribute with the target item p. We call this filtering strategy as **category filtering**. For an item node, we simply randomly sample another number of its user neighbor nodes. For efficient, we add the category attribute as an index of nodes to our graph adjacency to achieve the sampling process. Figure 1(b) shows an illustration of the-first-three-orders neighbors for user u_1 according to the example graph in Figure 1(a).

4.2.3 Backward Node Representation aggregation. After obtaining the K-order neighbors for user u and item p in the input sample $\mathbf{x}$, we aggregate the information of these neighbors backward to refine the representations of u and p. We use $e_u^{(K)}$ and $e_p^{(K)}$ to denote their refined representations after K layers of aggregation. We refine the node representation by aggregating the corresponding neighbors' representations from the K-th order to the 1-st order recursively, *i.e.,* the outputs of previous aggregation layers are the inputs of the next. The recursive calculation of the l-th (shown in the superscript, and $1 \leq l \leq K$) aggregation layer for node v, which can be either a user node u or an item node p, is given as follows:

$$e_n^{(l)} = Transformer(Q_n^{(l-1)}, K_n^{(l-1)}, V_n^{(l-1)}),\ \forall n \in \mathcal{N}_v^{K-l} \tag{7}$$

and the *Query*, *Key* and *Value* of the tansformer unit is given as:

$$Q_n^{(l-1)} = (e^I || e_n^{(0)}) W^I \tag{8}$$

$$K_n^{(l-1)} = (e_n^{(0)}, e_{m_1}^{(l-1)}, \cdots, e_{m_t}^{(l-1)}) \tag{9}$$

$$V_n^{(l-1)} = K_n^{(l-1)} \tag{10}$$

$$e_n^{(0)} = e_{pre(n)} || e_{q_{n,pre(n)}} || e_n \tag{11}$$

where $e_n^{(l)}$ is the refined representation of node n after the l-th aggregation layer; $e^I = e_u || e_q || e_p$ denotes the **intention representation** of the input sample $\langle u, q, p \rangle$; $e_n^{(0)}$ denotes the **upstream-enhanced** representation of the central node n, where $pre(n)$ is the precursor node of n, $q_{n,pre(n)}$ is the query on the edge between n and $pre(n)$; W^I is the translation matrix fusing the target intention and center node information; $K_n^{(l-1)}$ is a sequence containing the neighbourhood information to be aggregated, where $m_i \in \mathcal{N}_v^{K-l+1} \wedge pre(m_i) = n, 1 \leqslant i \leqslant t$. Note that 0-th order neighbor of a node v is itself: $\mathcal{N}_v^0 = \{v\}$. So when $l = K$, Equation (7)

gives us the K-th order representation of node v: $e_v^{(K)}$, which is the final representation of node v that we need.

In Equation (9), the neighborhood sequence $K_n^{(l-1)}$ has two parts: (1) the upstream-enhanced representation of central node $e_n^{(0)}$; (2) the refined representation of node n's neighbors $e_{m_i}^{(l-1)}$, generated from the previous aggregation layer. We sort neighbors in $K_n^{(l-1)}$ according to the click occurrence timestamps. As a result, the neighbors in the sequence $K_n^{(l-1)}$ are ordered according to timestamps of their clicks. So with the positional encoding [25], the transformer can learn time-series information in the higher-order connections.

Figure 2 illustrates the aggregation process for the user node u_1, assuming that $\langle u_1, q_{1,2}, p_2 \rangle$ is the input sample to be predicted. The right part of Figure 2 shows the details of the 1-st aggregation layer, *i.e.,* aggregating the 3-rd order neighbors information to refine the representation of the 2-nd order node u_3. Then the transformer output $e_{u_3}^{(1)}$ as the refined representation of node u_3, which will become the input of the next aggregation layer. Further, the information of the 2-nd order neighbors will propagate to refine the representation of node p_4. Finally, the representation of node u_1 will be updated.

4.3 Prediction Layer and Optimization

With the input sample $x = \langle u, q, p \rangle$ to be inferred, after K layers of aggregation, we finally get the refined representations $e_u^{(K)}$ and $e_p^{(K)}$. Then we concatenate them with their representation vectors, to obtain the overall representation for the input sample:

$$e_{\mathbf{x}} = e_u || e_q || e_p || e_u^{(K)} || e_p^{(K)} \tag{12}$$

We feed $e_{\mathbf{x}}$ into a Multi-Layer Perceptron (MLP) to predict click-through rate. Two different MLPs are adopted to predict for search and recommendation samples respectively:

$$\hat{y}_{\mathbf{x}} = \begin{cases} sigmiod(MLP_s(e_{\mathbf{x}})), & if\ \mathbf{x} \in \mathcal{S} \\ sigmiod(MLP_r(e_{\mathbf{x}})), & if\ \mathbf{x} \in \mathcal{R} \end{cases} \tag{13}$$

where $\mathcal{S}$ is the set of search samples; $\mathcal{R}$ is the set of recommendation samples; $\hat{y}_x$ is the predicted CTR for the sample $\mathbf{x}$. We employ the point-wise binary cross-entropy loss in our model:

$$\mathcal{L} = \sum_{\mathbf{x} \in \mathcal{S} \cup \mathcal{R}} y_{\mathbf{x}}\, log(\hat{y}_{\mathbf{x}}) + (1 - y_{\mathbf{x}})\, log\,(1 - \hat{y}_{\mathbf{x}}) \tag{14}$$

where $y_{\mathbf{x}} \in \{0, 1\}$ is the label of sample $\mathbf{x}$.

4.4 Discussion

Here we compare the aggregator of our SRJGraph model with those of existing GNN-based methods. Firstly, as we set the numbers of sampled k-th order neighbors to a fixed size S_k, the space and time complexity of $\mathcal{N}_u^k$ is fixed at $O(\prod_{i=1}^{k} S_i)$. So the complexity of our model is of the same order as scalable GraphSAGE [10] model. Then distinct from the aggregators used in traditional GNN [10, 15, 26] or graph-based models [12, 29, 35], our aggregation layer is intention- and upstream-aware:

(1) In existing GNN models, each aggregation layer only takes advantage of the local information within the current order. However, we argue that the information of the input sample $\langle u, q, p \rangle$ should be also considered in aggregation because it is useful for weighting neighbors' relative importance. So we encode the intention representation $e^I = e_u || e_q || e_p$ into the *Query* of the transformer unit in the aggregation layer. In this way, more attention will be put on neighbors that are more relevant to the target input sample that need to be predicted.
(2) For a node n on the graph, we represent its upstream-enhanced representation as $e_n^{(0)} = e_{pre(n)} || e_{q_{n,pre(n)}} || e_n$, which combines the upstream information $e_{pre(n)}$ and $e_{q_{n,pre(n)}}$. This upstream information explicitly recorded the precursor along the path of aggregation. It helps the model to dynamically weight the current neighbors' importance with more global information.

This intention- and upstream-aware aggregation layer has longer "vision" than traditional aggregation layer, and it dynamically learns the attention weights of high-order relations for different target samples.

5 EXPERIMENTS

In this section, we empirically evaluate SRJGraph on a real-world search and recommendation dataset [1]. We first describe the dataset and the CTR prediction baselines used in our experiment. Then we present the experiment results and detailed analysis. We aim to address the following research questions:

- **RQ1**: How does SRJGraph perform compared to existing CTR prediction models?
- **RQ2**: Does joint learning of search and recommendation improve the performance of CTR prediction?
- **RQ3**: What is the contribution of the intention and upstream-aware aggregation layer to the prediction performance?
- **RQ4**: What is the impact of aggregation depth and width on the performance of SRJGraph?

5.1 Dataset

The dataset used in this study is collected from *Taobao.com*, the largest e-commerce platform in China, containing 24 days of search and recommendation log data in September 2020. The dataset consists of two sub-datasets: Dataset-S and Dataset-R, corresponding to the search and recommendation data respectively. Each sample in the dataset contains a user, an item, a binary click label, and a timestamp of when this item is presented to the user. In Dataset-S, each sample also contains a search query. Besides, a user also has two additional features: gender and age, while an item has four additional features: brand, shop, category, and price. Each search query is segmented into terms. Real-value features, such as item price, are discretized into several levels. Therefore, all features are discrete and can be encoded using feature IDs. The detailed statistics for the dataset are shown in Table 1. Table 2 shows all features in our model, including user, item, and query-level features. And it is the first dataset constructed for the joint learning of search and recommendation CTR prediction in e-commerce, to the best of our knowledge. So we conduct the following experiments only on this dataset. We divide our dataset into four parts: background,

[1] Our codes can be obtained from https://www.github.com/XXX. Because the dataset used in this study is regulated by Alibaba group's policies, the application for it needs to be strictly reviewed. Please contact the corresponding author by email if you are interested in the dataset.

Table 1: The statistics of the dataset used in this study.

Dataset	#users	#queries	#items	#samples
Dataset-S	82,887	1,238,181	8,985,339	51,538,399
Dataset-R	82,887	-	2,425,443	23,663,828
Total	82,887	1,238,181	9,621,325	75,202,227

Table 2: The overview of user, item and query-level features. #distinct is the number of distinct sparse feature values.

Level	Feature	Type	#distinct	Dimension
user	id	sparse	82,888	16
	gender	sparse	3	4
	age level	sparse	16	4
item	id	sparse	9,621,326	16
	brand	sparse	169,267	8
	seller	sparse	722,971	8
	category	sparse	737	8
	price level	sparse	101	8
query	term	sparse sequence	98,404	16

training, validation and test. The background part spans the first 14 days, which provides historical user behaviors for CTR prediction models. The training, validation and test set spans the next seven, one and two days, respectively.

5.2 Baselines

In this study, we compare SRJGraph with the following baseline models:

Classical baseline:

- **DNN**: This model consists of the embedding layer and the prediction layer as introduced in Section 4. The prediction layer takes the representation vectors of the user, item, and query features from the embedding layer as inputs. This is a vanilla deep learning model for CTR prediction.
- **NFM** [11]: This model was proposed to overcome the linearity of factorization machines. It combines factorization machines and non-linear neural network to better capture higher-order feature interactions.
- **xDeepFM** [19]: This model consists of a compressed interaction network (CIN) and a MLP for prediction, which generates feature interactions in an explicit manner.

User behavior sequence baseline:

- **TEM** [4]: This model utilizes a transformer-based layer and takes the representations of search queries and items in the user behavior history as the inputs, to mine the user's preferences and search intents.
- **AEM+DIN** [2, 40]: This model is a combination of AEM [2] and DIN [40]. AEM performs an attention operation between the user's historical behavior sequence and the current search query. DIN performs a similar attention operation between the user's historical behavior sequence and the current target item. We concatenate the representation of the search query and the target item as the query of the attention operation as AEM+DIN. And it performs better than AEM and DIN. So we only reports the results of AEM+DIN.

Table 3: Overall comparisons with the baselines.

Model	Search		Recommendation	
	AUC	RelaImpr	AUC	RelaImpr
DNN	0.6212	-	0.6487	-
NFM	0.6215	0.25%	0.6493	0.40%
xDeepFM	0.6218	0.50%	0.6502	1.01%
TEM	0.6222	0.83%	0.6511	1.61%
AEM+DIN	0.6230	1.49%	<u>0.6519</u>	2.15%
GAT	0.6318	8.75%	0.6514	1.82%
HGT	0.6321	8.99%	0.6515	1.88%
NGCF	0.6313	8.33%	0.6513	1.75%
LightGCN	<u>0.6325</u>	9.32%	0.6513	1.75%
SRJGraph	**0.6376***	**13.53%**	**0.6554***	**4.51%**

*: significantly outperforms the <u>runner-up</u> based on paired t-test at the significance level of 0.001.

GNN baseline:

- **GAT** [26]: This GNN model adopts the attention mechanism to weight the importance of neighbors during aggregation, and learn node representations for tasks end-to-end.
- **HGT** [15]: This GNN model designs edge-type and node-type dependent parameters to deal with different relations and learn the heterogeneous node representations.
- **NGCF** [29]: This model adopts a user-item interaction graph to better learn users' interests in the recommender system. It utilizes the high-order connectivity relations between users and items.
- **LightGCN** [12]: To better adapt GNN to the recommendation task, this model discards feature transformation and nonlinear activation in NGCF. The user and item embeddings are linearly propagated on the user-item interaction graph. Therefore, this model becomes more concise and efficient.

Joint learning baseline:

- **JSR** [36]: A framework to joint learn a search model and a recommendation model, which can be applied to the above baselines as shown in their original paper.

These baselines cover from recommendation and search CTR prediction models to graph neural models. NFM, xDeepFM, DIN, NCGF and LightGCN are proposed for recommendation, AEM, TEM are proposed for search, and GAT, HGT are the state-of-the-art graph neural networks. Meanwhile, by adding search query features into prediction layer, recommendation models are also applicable to the search CTR prediction task, and vice versa. Therefore, these baseline models are adaptive to both recommendation or search CTR prediction tasks. All models are trained using point-wise cross-entropy loss, which perform no worse than the pair-wise loss used in some of the original baselines.

5.3 Experimental Setup

5.3.1 Model Settings. We implemented all models using TensorFlow [1]. The optimizers for all models are chosen to be Adagrad [8], which performs the best in our experiments. We tune hyper-parameters of all models and adopt the early-stop strategy based on their performance on the validation set. On our SRJGraph, the average degrees of user and item nodes are 173 and 4. So we sample

Table 4: Effect of joint learning on search and recommendation data.

Joint Model	Search			Recommendation		
	AUC	RelaImpr		AUC	RelaImpr	
		vs. Non-joint	vs. JSR-DNN		vs. Non-joint	vs. JSR-DNN
JSR-DNN	0.6240	2.31%	-	0.6503	1.08%	-
JSR-NFM	0.6244	2.49%	0.32%	0.6503	0.67%	0.00%
JSR-xDeepFM	0.6255	3.04%	1.21%	0.6518	1.07%	1.00%
JSR-TEM	0.6260	3.11%	1.61%	0.6528	1.13%	1.66%
JSR-AEM+DIN	0.6268	3.09%	2.26%	0.6539	1.32%	2.40%
SRJGraph-GAT	0.6348	2.28%	8.71%	0.6532	1.19%	1.93%
SRJGraph-HGT	0.6346	1.89%	8.55%	0.6534	1.25%	2.06%
SRJGraph-NGCF	0.6341	2.13%	8.15%	0.6537	1.59%	2.26%
SRJGraph-LightGCN	0.6358	2.26%	9.52%	0.6539	1.72%	2.40%
Our SRJGraph	**0.6376***	-	10.97%	**0.6554***	-	3.39%

*: significantly outperforms the runner-up based on paired t-test at the significance level of 0.001.

more neighbors for a user node than for an item node. For the aggregation layer on SRJGraph, we set the highest-order of aggregation K to 3. At last the numbers of sampled neighbors in the 3 orders are [25, 1, 10] for user node, and [1, 25, 1] for item node. During joint learning, our model randomly selects a mini-batch of samples from Dataset-S or Dataset-R at a time. We sample neighbors as described in Section 4.2.2 for all GNN models, as it performs better than no category filtering. For the model TEM and AEM+DIN, the user behavior sequence is 25. We independently run 5 times of experiment for each model with different random seeds to report the average results and compute the significance level.

5.3.2 Evaluation Metrics. AUC [9] is a widely used metric in CTR prediction [20, 32, 40] for its ability to handle label imbalance in CTR dataset. In this study, we also use AUC to evaluate models' performance. To calculate the relative improvement over baseline methods, we follow [33] to use *RelaImpr*. The relative improvement of model A over model B measured by *RelaImpr* is given as:

$$RelaImpr = (\frac{AUC_A - 0.5}{AUC_B - 0.5} - 1) \times 100\% \quad (15)$$

5.4 Comparison with Baselines (RQ1)

To address **RQ1**, we compare the performance of all methods and report the results in Table 3. We find the following observations:

- DNN performs the worst among all models. This indicates that a multi-layer perceptron is insufficient for CTR prediction.
- In user behavior sequence baselines, AEM+DIN outperforms TEM in both search and recommendation tasks. This indicates that using the information of user, query and item together as the *Query* in the attention unit of AEM+DIN better activates the user behavior sequence than only using the information of query as attention *Query* in TEM.
- GNN baselines GAT, HGT, NGCF and LightGCN achieve high prediction performance on both search and recommendation tasks, indicating the effectiveness of GNN.
- Our SRJGraph model performs the best in both tasks. In search task, SRJGraph outperforms the best baseline LightGNN by 0.51% in AUC score. In recommendation task, SRJGraph outperforms the best baseline AEM+DIN by 0.35% in AUC score. And t-test results show that these improvements are significant.

5.5 Effect of Joint Learning (RQ2)

Our SRJGraph model uses a GNN to perform joint learning on search and recommendation data. In this section, we perform experiments to see how baseline models benefit from joint learning framework JSR [36], and compare the Joint-models with our SRJGraph model.

After applying the JSR framework to the baseline models, they can be trained on both datasets. To be more specific, for classical baselines, joint learning is achieved by sharing embedding tables and training on both search and recommendation datasets. For user behavior sequence baselines, user's historical behaviors are further collected from both datasets. For GNN baselines, the user-item graph is also constructed on both search and recommendation interaction as our SRJGraph. Table 4 presents the results of our experiment. In Table 4, "vs. Non-joint" columns correspond to the relative improvement of JSR-models compared to single-task-learning models. Because SRJGraph is a joint learning model in nature, it has no single-task-learning results and thus no result in the "vs. Non-joint" columns. The "vs. JSR-DNN" columns correspond to the relative improvement for each joint-learning model over JSR-DNN. From Table 4, we have the following findings:

- The results in "vs. Non-joint" columns show that all baseline models gets improved by joint learning, on both search and recommendation tasks. This further validates our idea that joint learning alleviates the data sparsity problem and improves performance of these CTR prediction models.
- Compared to the jointly learned baselines, our SRJGraph model still achieves the best performance on both search and recommendation tasks. And the improvement of SRJGraph over the joint-learning baselines is statistically significant. This shows the effectiveness of our intention and upstream-aware aggregation mechanism in SRJGraph model.

5.6 Study of SRJGraph (RQ3 & RQ4)

In this section, we further conduct an ablation study to investigate the impact of the intention and upstream-aware aggregation operation. We also investigate the effectiveness of query as an attribute of edges in our SRJGraph. Finally, we analyze the impact of aggregation depth and width in SRJGraph.

Table 5: Performance of models in the ablation study, where "w/o" is short for "without".

Model	Search		Recommendation	
	AUC	RelaImpr	AUC	RelaImpr
SRJGraph	**0.6376**	-	**0.6554**	-
SRJGraph w/o query	0.6373	-0.22%	0.6547	-0.45%
SRJGraph w/o intention	0.6364	-0.87%	0.6550	-0.26%
SRJGraph w/o upstream	0.6368	-0.58%	0.6540	-0.90%

5.6.1 Ablation Study (RQ3). Here we list the variants of SRJGraph to be compared in the ablation study:

- **SRJGraph without query**: In this model, we mask all search queries on the graph (not the intention representation of Query input used in the transformer). We want to examine the effect of queries on our heterogeneous attributed graph by comparing this model with the original SRJGraph.
- **SRJGraph without intention**: In this model, we mask the intention representation of the input sample $\langle u, q, p \rangle$, to examine the effect of our intention-aware aggregation operation.
- **SRJGraph without upstream**: In this model, we modify the aggregation layer by masking upstream information in the representation of central node. This model is to examine the effect of our upstream-aware aggregation operation.

The results of ablation study is presented in Table 5. And we have the following findings:

- *SRJGraph without query* performs worse on both search and recommendation than the original SRJGraph model. This indicates that queries on the edges of the user-item interaction graph are important to the effectiveness of SRJGraph in jointly modeling the user-item relations in search and recommendation. Without query, not only the search performance but also the recommendation performance deteriorates.
- *SRJGraph without upstream* and *SRJGraph without intention* are consistently worse than the original SRJGraph model. This indicates that the intention and upstream-aware aggregation layer is effective in learning the attention weights according to more global information of the input sample and the aggregation path. These results prove the effectiveness of our intention and upstream-aware aggregation layer.

5.6.2 Effect of Aggregation Depth and Width (RQ4). We define aggregation depth to be the highest order of aggregation in SRJGraph. And we define aggregation width to be the numbers of neighbors sampled at each order of aggregation. So the aggregation width is actually a list of numbers. As mentioned in section 5.3.1, we sample more neighbors for a user node than an item node. The default aggregation width setup is "(25,10)" for a user node, and "1" for an item node. If the aggregation depth is 4, then the aggregation width is [25, 1, 10, 1] for the target user and [1, 25, 1, 10] for the target item.

Table 6 shows the model performance with different aggregation depths. In this experiment, we varied aggregation depth from 1 to 4, while fixing the aggregation width setup for a user node to "(25,10)". We find that when the aggregation depth increases from 1 to 3, the performance on both search and recommendation improves.

Table 6: Effect of different aggregation depths.

Depth	Search		Recommendation	
	AUC	RelaImpr	AUC	RelaImpr
1	0.6293	-	0.6551	-
2	0.6375	6.34%	0.6553	0.13%
3	**0.6376**	**6.42%**	**0.6554**	**0.19%**
4	0.6375	6.34%	0.6548	-0.19%

Table 7: Effect of different aggregation widths.

Width	Search		Recommendation	
	AUC	RelaImpr	AUC	RelaImpr
(10,4)	0.6365	-	0.6552	-
(15,6)	0.6368	0.22%	0.6552	0.00%
(20,8)	0.6374	0.66%	0.6554	0.13%
(25,10)	**0.6376**	**0.81%**	**0.6554**	**0.13%**

But the performance drops when aggregation depth increases to 4. Larger aggregation depth collects information from neighbors further away. When the aggregation depth gets too large, distant and less relevant nodes are aggregated, leading to poorer performance.

Table 7 reports the model performance with different aggregation widths. In this experiment, we varied the setup of aggregation width, while fixing the aggregation depth to 3. The aggregation width for an item node is always 1. The width setup "(10,4)" means the aggregation width is [10, 1, 4] for the target user and [1, 10, 1] for the target item. We find that as the aggregation width increases, the model performance also increases. However, the computation cost of training and inference will also increase dramatically. In practice, we need to balance the performance boost and computation cost.

6 CONCLUSION AND FUTURE WORK

In this paper, we use a GNN-based method for joint learning of the CTR prediction task in e-commerce search and recommendation. In this study, we first propose a novel heterogeneous attributed graph to model the user-item interactions for both search and recommendation. Then, we design a novel GNN-based model named SRJGraph for the CTR prediction task. Besides modeling the local information of the central node and its neighbors, we also incorporate the information of the input sample (**intention**) and the precursor relation (**upstream**) into the aggregation layer of SRJGraph. Further, we systematically conduct a series of experiments to analyze the performance of our proposed SRJGraph model. Experiment results show that SRJGraph is superior to the most advanced baseline methods in CTR prediction. We also show that the joint learning CTR prediction for search and recommendation is beneficial for both scenarios. Our ablation study shows that the intention and upstream-aware aggregation layer significantly improves model performance. However, there are also limitations. In this work, we simply use a sum-pooling operation on the embeddings of query terms to represent the query. In the future, recurrent neural networks or transformer models will be explored to learn more from the query attribute.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI'16*. USENIX Association, 265–283.

[2] Qingyao Ai, Daniel N Hill, SVN Vishwanathan, and W Bruce Croft. 2019. A zero attention model for personalized product search. In *CIKM'19*. 379–388.

[3] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835* (2018).

[4] Keping Bi, Qingyao Ai, and W Bruce Croft. 2020. A Transformer-based Embedding Model for Personalized Product Search. In *SIGIR'20*. 1521–1524.

[5] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR'17*. 335–344.

[6] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. 2018. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *WWW'18*. 639–648.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS'16*. 3844–3852.

[8] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, 7 (2011).

[9] Tom Fawcett. 2006. An introduction to roc analysis. *Pattern Recognition Letters* 27, 8 (2006), 861 – 874.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS'17*. 1024–1034.

[11] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR'17*. ACM, 355–364.

[12] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *SIGIR'20*. ACM, 639–648.

[13] Xiangnan He, Ming Gao, Min-Yen Kan, and Dingxian Wang. 2016. Birank: Towards ranking on bipartite graphs. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 57–71.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW'17*. 173–182.

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 2704–2710.

[16] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P Xing. 2019. Rethinking knowledge graph propagation for zero-shot learning. In *CVPR'19*. 11487–11496.

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

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[19] 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 *KDD'18*. ACM, 1754–1763.

[20] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature generation by convolutional neural network for click-through rate prediction. In *WWW'19*. ACM.

[21] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: A view from the trenches. In *SIGKDD'13*. ACM, 1222–1230.

[22] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.

[23] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. 2018. A taxonomy of queries for e-commerce search. In *SIGIR'18*. 1245–1248.

[24] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. 2018. User intent, behaviour, and perceived satisfaction in product search. In *WSDM'18*. 547–555.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS'17*. 5998–6008.

[26] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR'18*. OpenReview.net.

[27] Jian Wang, Yi Zhang, and Tao Chen. 2012. Unified Recommendation and Search in E-Commerce. In *Information Retrieval Technology, 8th Asia Information Retrieval Societies Conference, AIRS 2012. Proceedings*, Vol. 7675. Springer, 296–305.

[28] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item silk road: Recommending items from information domains to social users. In *SIGIR'17*. 185–194.

[29] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR'19*. ACM, 165–174.

[30] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW'19*. 2022–2032.

[31] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* (2020).

[32] Weinan Xu, Hengxu He, Minshi Tan, Yunming Li, Jun Lang, and Dongbai Guo. 2020. Deep interest with hierarchical attention network for click-through rate prediction. In *SIGIR'20*. ACM, 1905–1908.

[33] Ling Yan, Wu-Jun Li, Gui-Rong Xue, and Dingyi Han. 2014. Coupled group lasso for web-scale ctr prediction in display advertising. In *ICML'14*. JMLR.org, 802–810.

[34] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: High-order proximity for implicit recommendation. In *RecSys'18*. 140–144.

[35] 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 *SIGKDD'18*. ACM, 974–983.

[36] Hamed Zamani and W. Bruce Croft. 2018. Joint Modeling and Optimization of Search and Recommendation. In *DESIRES'18*. CEUR-WS.org, 36–41.

[37] Hamed Zamani and W. Bruce Croft. 2020. Learning a joint search and recommendation model from user-item interactions. In *WSDM'20*. ACM, 717–725.

[38] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR meets graph embedding: A ranking model for product search. In *WWW'19*. ACM, 2390–2400.

[39] 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 *AAAI'19*. AAAI Press, 5941–5948.

[40] 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 *KDD'18*. ACM, 1059–1068.