Leveraging Bidding Graphs for Advertiser-Aware Relevance Modeling in Sponsored Search

Shuxian Bi[1]*, **Chaozhuo Li**[2]†, **Xiao Han**[2], **Zheng Liu**[2], **Xing Xie**[2],
Haizhen Huang[2], **Zengxuan Wen**[2]
[1]University of Science and Technology of China
[2]Microsoft
stanbi@mail.ustc.edu.cn
{cli, xiaoha, zhengliu, xingx, hhuang, zewen}@microsoft.com

Abstract

Recently, sponsored search has become one of the most lucrative channels for marketing. As the fundamental basis of sponsored search, relevance modeling has attracted increasing attention due to the tremendous practical value. Most existing methods solely rely on the query-keyword pairs. However, keywords are usually short texts with scarce semantic information, which may not precisely reflect the underlying advertising intents. In this paper, we investigate the novel problem of advertiser-aware relevance modeling, which leverages the advertisers' information to bridge the gap between the search intents and advertising purposes. Our motivation lies in incorporating the unsupervised bidding behaviors as the complementary graphs to learn desirable advertiser representations. We further propose a **B**idding-**G**raph augmented **T**riple-based **R**elevance model BGTR with three towers to deeply fuse the bidding graphs and semantic textual data. Empirically, we evaluate the BGTR model over a large industry dataset, and the experimental results consistently demonstrate its superiority.

1 Introduction

Large commercial search engines typically provide organic web results in response to user queries and then supplement with sponsored ads. Advertisers can bid on keywords so that their ads show up when people are looking for exactly the kind of things they sell. As the fundamental component of sponsored search systems, relevance model measures the semantic closeness between an input query and a candidate keyword, which is capable of improving the user experience and driving revenue for the advertisers (Ling et al., 2017).

Existing relevance models can be roughly categorized into two sets: one-tower and two-tower structures. One-tower structure learns the joint embedding of the concatenated query-keyword text, while the two-tower structure generates the query embedding and keyword embedding separately. The core components are query/keyword encoders, which are implemented as the powerful Natural Language Understanding (NLU) models to capture the semantic correlations inside the query-keyword pairs.

Query	Keyword	Advertiser	Label
apple pie merchant	apple pie	target.com	Good
apple pie merchant	apple pie	delish.com	Bad
salt water fishing	salt water fish	amazon.com	Good
salt water fishing	salt water fish	petco.com	Bad

Table 1: Two pairs of representative examples to show same query-keyword pair may have different relevance meanings given different advertisers.

Although SOTA relevance models achieve impressive performance on the off-line evaluation, the complaints from advertisers are constantly emerging on the on-line industry platform. Based on the complaints collected by a popular commercial search engine, we found that the main reason lies in that the bid keywords may not precisely reflect the advertising purposes. To attract more interests and traffics from the search engine, advertisers may bid the shorten or unstructured texts as the keywords. Table 1 shows two pairs of representative examples. Within the first example, two advertisers "target.com" and "delish.com" both bid the keyword "apple pie". Given an input query "apple pie merchant", relevance models select keyword "apple pie" based on the semantic closeness and display ads from these two advertisers on the search result page. However, "delish.com" is a recipe website providing cooking guides instead of selling foods, leading to the mismatch between the search intent and advertising purpose. Similar issue happens in the second example as petco.com, a pet store, which does not sell the fishing tools. Same query-keyword pairs may have different relevance

* Work is done during internship at Microsoft Research Asia.

† Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2215–2224
November 7–11, 2021.

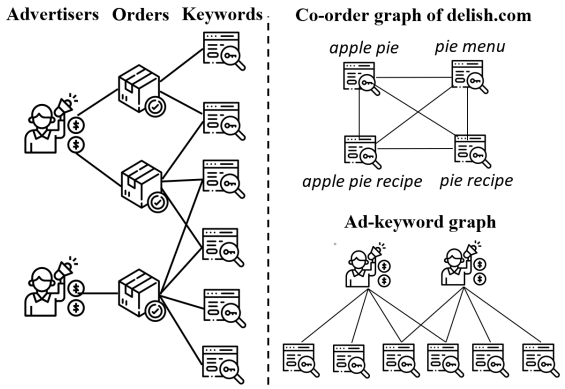

Figure 1: The illustration of the bidding graphs.

meanings given different advertisers. Thus, it is crucial to explore the information of advertisers to better understand the underlying advertising purpose and bridge the gap between the search and advertising intents.

Different from traditional relevance models which solely rely on the query-keyword pairs, in this paper we investigate the novel problem of triple-based (i.e., query-keyword-advertiser) relevance modeling. Two critical challenges need to be addressed. Firstly, existing approaches usually learn the keyword representations by encoding the semantic text, which obtain identical embeddings for the same keywords. However, as discussed above, same keywords may have different meanings given different advertisers (e.g., keyword "apple pie" bid by "delish.com" is more similar to "apple pie recipe"). Hence, the keyword representations should be advertiser-aware. Secondly, how to learn the desirable representations for the advertisers is not straight-forward. Information from the domain URL is too obscure to indicate the intrinsic features of the advertisers (e.g., it is troublesome to interpret the URL "indeed.com " as a job seeking website literally). The homepages are full of various HTML elements and commodities, and extracting useful information from such noisy data is non-trivial. External knowledge graph (e.g., Freebase) may also not be a good solution as many small businesses are not included, leading to the comparatively lower coverage rate.

In this paper, we propose to leverage the bidding behaviors of advertisers to learn the quality representations beyond the semantic texts. As shown in the left part of Figure 1, orders are placed by advertisers to the search engine, which contain a set of keywords belonging to the same category. Three components (advertisers, orders and keywords) can naturally form two types of bidding graphs: the co-order graph and the ad-keyword graph. For each advertiser, we construct a homogeneous co-order graph, in which nodes are the keywords bid by this advertiser and the edges denote the co-order relationships. These co-order graphs facilitate the learning of advertiser-aware keyword representations. For example, as shown in Figure 1, with the co-order keywords "apple pie menu" and "pie recipe", we can understand the keyword "apple pie" bid by "delish.com" refers to recipes. The ad-keyword graph is a bipartite graph contains two types of nodes: advertisers and keywords, in which nodes are connected by the bidding behaviors. Our insight lies in the phenomenon of homophily as advertisers with similar bid keywords are also tend to be similar, which can be leveraged to learn quality advertiser representation with high converge rate. Based on these observations, we further propose a **B**idding-**G**raph augmented **T**riple-based **R**elevance model BGTR, which includes three towers: the query encoder, the keyword encoder and the advertiser encoder. BGTR model is capable of deeply fusing the semantic textual information and the bidding graphs. Experimental results on the large industry dataset demonstrate that our proposal can effectively improve the performance of relevance modeling.

We summarize the main contributions of this paper as follows.

- We study the novel problem of advertiser-aware relevance modeling, which is a critical challenge in the industry area but rarely explored yet.
- We propose to leverage the bidding graphs as complementary to enrich the semantic information. A triple-based model BGTR is proposed to effectively fuse textual data and bidding graphs.
- Extensively, we evaluate our proposed model on a large industry dataset. Experimental results demonstrate the superior performance of the proposed BGTR model.

2 Problem Definition

In this section, we will formally define the studied problem. Different from traditional query-keyword based methods, here we introduce the definition of "advertiser" to form up the triple:

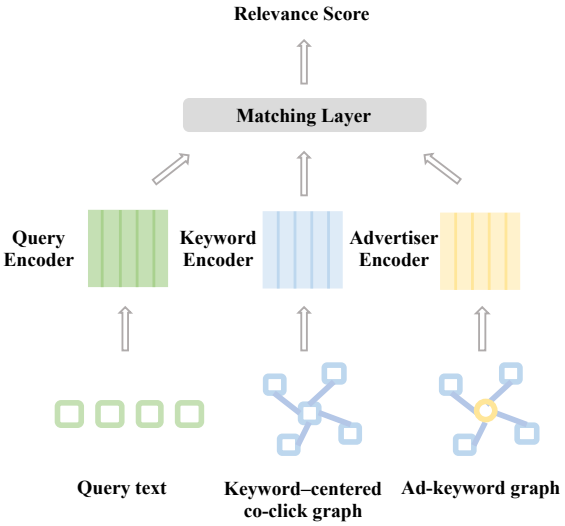

Figure 2: The framework of BGTR model.

$\mathbf{Z} = \{< q_i, k_i, a_i >\}$, in which q_i denotes the input query, k_i denotes the keyword and a_i is an advertiser who bid the keyword k_i. For each advertiser a_i, its corresponding co-order graph is defined as $\mathbf{O}_i = \{\mathbf{K}_i, \mathbf{E}_i\}$, in which $\mathbf{K}_i$ denotes the set of keywords bid by a_i. $\mathbf{E}_i \in \mathbb{N}^{|\mathbf{K}_i|\times|\mathbf{K}_i|}$ is the adjacency matrix, which includes the co-order relationships between different keywords. The ad-keyword graph is defined as a bipartite graph: $\mathbf{G} = \{\mathbf{A}, \mathbf{K}, \mathbf{E}\}$, in which $\mathbf{A}$ and $\mathbf{K}$ denote the whole set of advertisers and keywords, respectively. $\mathbf{E} \in \mathbb{N}^{|\mathbf{A}|\times|\mathbf{K}|}$ is the adjacency matrix, which includes the bidding signals between advertisers and keywords. We aim to learn a classifier f: $(q_i, k_i, a_i) \rightarrow \{0, 1\}$ by fusing the ground truth set and the bidding graphs $\mathbf{O}_i$ and $\mathbf{G}$.

3 Methodology

3.1 Framework

Figure 2 exhibits the framework of the proposed BGTR model, which is an extension of the two-tower models (e.g., C-DSSM (Gao et al., 2015) and TwinBERT (Lu et al., 2020)). The embeddings of query, keyword and advertiser are learned separately. As there exist millions of candidate keywords and advertisers, it is impracticable to use a single text encoder (e.g., BERT) to compute the similarity between a search query and each keyword-advertiser pair one-by-one (Lu et al., 2020). Hence, the triple-tower structure is a feasible choice for online serving as we could precompute the keyword and advertiser representations in advance. When a query comes, we can easily generate its embedding and calculate the similarities between the input query embedding and cached representations of keywords and advertisers.

3.2 Query Encoder

Query encoder aims to learn the quality representation for the input query q_i to capture the search intents accurately. Because queries are input by the search engine users and irrelevant to the bidding behaviors, query encoder solely relies on the semantic texts inside the input query, which can be implemented as any layer-wise text encoding models. Here we select the powerful BERT model as the query encoder. The input query is first tokenized using the BERT WordPiece tokenizer (Wu et al., 2016). For each token within the input sequence, the initial embedding is acquired with the summation of its token embedding and positional embedding. Then, these initial embeddings are fed into the transformer encoder layers to obtain a sequence of embedding vectors corresponding to the tokens in the input query. Finally, we take the final hidden vector of [CLS] token as the final query representation following TwinBERT model.

3.3 Advertiser-aware Keyword Encoder

Traditional keyword encoders learn the representations solely rely on the text of the input keyword k_i. However, keywords are usually quite short with scarce semantic information, which are insufficient to precisely depict the advertising intents. Besides, the keyword representations should be advertiser-aware as discussed in the introduction section. Given the input tuple $< q_i, k_i, a_i >$, we propose to incorporate the co-order graph $\mathbf{O}_i$ of advertiser a_i as complementary information to learn quality advertiser-aware representation for the keyword k_i. On the one hand, keywords within the same order placed by an advertiser tend to depict the similar advertising intents, which can provide more abundant semantic information compared with a single sentence. On the other hand, given advertisers with different backgrounds, the co-order neighbors of the same keyword also tend to be different. Leveraging such advertiser-specific information can learn distinct representations for the same keyword bid by different advertisers.

Graph Neural Networks (GNNs) (Veličković et al., 2017; Hamilton et al., 2017) are widely applied on graph structural data with promising performance. In most GNN models, the node features

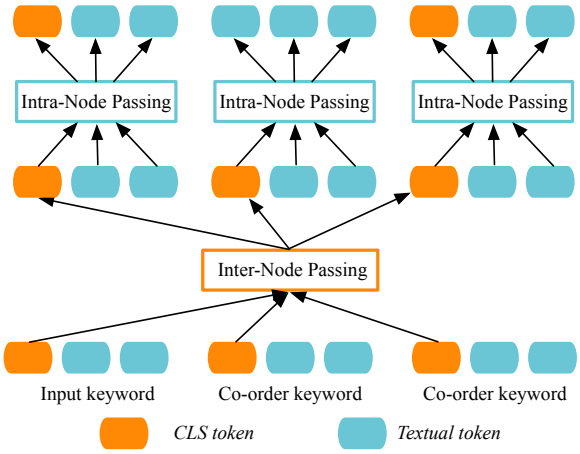

Figure 3: The illustration of a single layer in advertiser-aware keyword encoder.

are pre-trained and fixed in the training phase. Recently, several approaches are proposed to co-train both text encoders and GNN parameters to better fuse the textual data and graph topology (Zhu et al., 2021; Zhang et al., 2020; Li et al., 2021; Yang et al., 2021). Text in each node is firstly encoded into the a textual embedding vector through a multiple-layer NLU model, and then the textual embeddings of neighbor nodes are aggregated into the center node following the guidance of topology connections. This cascaded workflow is essentially a loosely coupled framework as the node can not make reference to its neighborhood while encoding its own textual feature, leading to the inferior node representations.

Here we aim to deeply fuse the semantics inside the keyword with its co-order neighborhood. Our insight lies in that in the text-encoding layers, each token can not only attend to other tokens within the center node, but also attend to tokens in its neighbors. We propose to utilize the embedding of a special token as the intermediate to efficiently pass messages between center node and its neighbors. Given the input keyword k_i and its neighbor set kn_i, the texts are tokenized using the BERT WordPiece tokenizer (Wu et al., 2016). After that, a [CLS] token is padded in the front of tokens in each sentence, whose embedding is viewed as the representation of the belonging sentence. As shown in Figure 3, each layer in the adaptive keyword encoder includes two components: intra-node passing and inter-node passing.

3.3.1 Inter-Node Passing

Inter-node passing aims to convey information among nodes through the co-order relations. Notation $\mathbf{m}_{ij}^{(l)}$ denotes the embedding of the j-th token in the i-th node in the layer l. Index i is set to 0 for the center node and $j = c$ means this is the embedding of [CLS] token. In the l-th layer, [CLS] token embeddings $\mathbf{m}_{ic}^{(l-1)}$ of all the nodes are firstly collected and gathered together as an inter-node matrix $\mathbf{M_c}^{(l-1)} \in \mathbb{R}^{(N+1)\times d_h}$, in which N is the number of neighbors and d_h denotes the dimension of latent embedding. Then, the multi-head graph attention is employed on the matrix $\mathbf{M_c}^{(l-1)}$ to exchange information between [CLS] embeddings of different nodes. For an arbitrary attention head, inter-node passing is defined as:

$$\hat{\mathbf{M}}_c^{(l-1)} = \text{softmax}\left(\frac{\mathbf{Q}_c^{(l-1)}\mathbf{K}_c^{(l-1)\top}}{\sqrt{d_h}}\right)\mathbf{V}_c^{(l-1)}, \tag{1}$$

where

$$\begin{cases} \mathbf{Q}_c^{(l-1)} &= \mathbf{M}_c^{(l-1)}\mathbf{W}_{Qc}^{(l-1)}, \\ \mathbf{K}_c^{(l-1)} &= \mathbf{M}_c^{(l-1)}\mathbf{W}_{Kc}^{(l-1)}, \\ \mathbf{V}_c^{(l-1)} &= \mathbf{M}_c^{(l-1)}\mathbf{W}_{Vc}^{(l-1)}. \end{cases} \tag{2}$$

in which matrices $\mathbf{W}_{Qc}^{(l-1)}, \mathbf{W}_{Kc}^{(l-1)}, \mathbf{W}_{Vc}^{(l-1)} \in \mathbb{R}^{d_h \times d_h}$ denote the trainable variables. The inter-node message passing allows the reciprocal interchange among the co-order keywords, which ensures the topological information is properly encoded into the generated [CLS] embeddings.

3.3.2 Intra-Node Passing

Then the topology-preserving [CLS] embeddings $\hat{\mathbf{m}}_{ic}^{(l-1)}$ in the generated matrix $\hat{\mathbf{M}}_c^{(l-1)}$ are dispatched to the corresponding nodes. For the node i, we can obtain a matrix $\hat{\mathbf{M}}_i^{(l-1)} = [\hat{\mathbf{m}}_{ic}^{(l-1)}, \mathbf{m}_{i0}^{(l-1)}, \cdots, \mathbf{m}_{in}^{(l-1)}]^\top \in \mathbb{R}^{(n+1)\times d_h}$ by combining the topology augmented [CLS] embedding $\hat{\mathbf{m}}_{ic}^{(l-1)}$ and the textual token embeddings $\mathbf{m}_{ij}^{(l-1)}$. Then, similar to the inter-node passing, we also employ the multi-head attentions on this matrix as follows:

$$\mathbf{M}_i^{(l)} = \text{softmax}\left(\frac{\mathbf{Q}_i^{(l-1)}\mathbf{K}_i^{(l-1)\top}}{\sqrt{d_h}}\right)\mathbf{V}_i^{(l-1)}, \tag{3}$$

where

$$\begin{cases} \mathbf{Q}_i^{(l-1)} &= \hat{\mathbf{M}}_i^{(l-1)}\mathbf{W}_{Q}^{(l-1)}, \\ \mathbf{K}_i^{(l-1)} &= \hat{\mathbf{M}}_i^{(l-1)}\mathbf{W}_{K}^{(l-1)}, \\ \mathbf{V}_i^{(l-1)} &= \hat{\mathbf{M}}_i^{(l-1)}\mathbf{W}_{V}^{(l-1)}. \end{cases} \tag{4}$$

in which matrices $\mathbf{W}_Q^{(l-1)}, \mathbf{W}_K^{(l-1)}, \mathbf{W}_V^{(l-1)} \in \mathbb{R}^{d_h \times d_h}$ denote the trainable variables in the l-th layer. A straight-forward strategy is to concatenate the texts from all the nodes as a long sentence, and then feed it into the BERT model. Such a long sentence will lead to the low efficiency of the BERT encoders. Also it is intractable to distinguish the tokens from the center node or its neighbors. In the intra-node passing phase, each textual token will attend to the topology-preserving [CLS] token, which means the semantic information from other nodes is also incorporated indirectly. In addition, [CLS] token also collects information from the textual tokens within the same node, which can be used in the inter-node passing phase of next layer.

Multiple layers of inter-node passing and intra-node passing are alternately deployed. The [CLS] embedding of the input keyword in the last layer is outputted as the final representation. Assume each node has s tokens and there exist t nodes. The attended field sizes of inter-node passing and intra-node passing are t and $s+1$ respectively, which is significantly less than the directly concatenating approach (each token will attend to $(t \times (s+1))$ tokens). This intermediate based structure ensures the adaptive keyword encoder not only can deeply fuse the textual information and co-order graph, but also can maintain the model efficiency.

3.4 Disentangled Advertiser Encoder

In this subsection, we will introduce the details of the advertiser encoder. Different from the straight-forward approaches like URL, homepage or external knowledge graph, the ad-keyword graph $\mathbf{G}$ is introduced to learn the desirable advertiser representations. The bipartite graph $\mathbf{G}$ contains two types of nodes (advertisers and keywords) and the bidding relationships among the nodes. Our motivation is that advertisers with similar bid keywords are also tend to be similar. As a single advertiser may bid thousands of keywords, this bidding graph can be very huge. It is infeasible to co-train the advertiser encoder along with other two towers. Here we propose to learn the advertiser embeddings based on the unsupervised link prediction task, and then view them as trainable embeddings in the downstream relevance modeling task.

Existing GNN models use weighted aggregation of neighborhood information as the enrichment to the center node. In the ad-keyword graph, advertisers may bid various keywords due to the great diversity of advertising intents. The bidding interactions are latently generated from highly sophisticated intent factors. Learning embeddings that reveal and disentangle these latent intent factors can enhance the expressiveness. Firstly, we formally define the disentangled representations. Assume there exist T latent factors, we expect that the learned embeddings of advertisers and keywords are composed of T independent components: $\mathbf{h}_a = [\mathbf{z}_{a,1}, \mathbf{z}_{a,2}, \cdots, \mathbf{z}_{a,T}]$ and $\mathbf{h}_k = [\mathbf{z}_{k,1}, \mathbf{z}_{k,2}, \cdots, \mathbf{z}_{k,T}]$. Each component measures the correlation between the t-th aspect of the advertiser or keyword and the t-th latent factor. As the advertiser embeddings are the learning targets, next we will introduce the learning details of $\mathbf{h}_a$.

The feature vector of advertisers are randomly initialized as $\mathbf{v}_a$. For the keywords, we utilize the efficient convolutional neural network (CNN) to learn the textual embedding $\mathbf{v}_k$ as BERT is too expensive to handle such a large number of short texts. Given an advertiser a along with one of her bid keywords k, we first use a projection matrix $\mathbf{W}_t$ to map these feature vectors into the t-th factor related subspace:

$$\begin{aligned} \mathbf{s}_{a,t}^{(0)} &= \frac{\sigma(\mathbf{W}_t^\top \mathbf{v}_a)}{||\sigma(\mathbf{W}_t \top \mathbf{v}_a)||_2} \\ \mathbf{s}_{k,t}^{(0)} &= \frac{\sigma(\mathbf{W}_t^\top \mathbf{v}_k)}{||\sigma(\mathbf{W}_t \top \mathbf{v}_k)||_2} \end{aligned} \tag{5}$$

in which the superscript 0 denotes the 0-th layer and σ is the activation function.

After that, in the l-th layer, we need to uncover the the probability $p_{a,k,t}$ that the advertiser a bids the keyword k due to the t-th factor, which is defined as follows:

$$p_{a,k,t}^{(l)} = \frac{\exp(\mathbf{s}_{a,t}^{(l)\top} \mathbf{s}_{k,t}^{(l)})}{\sum_{i=1}^{T} \exp(\mathbf{s}_{a,i}^{(l)\top} \mathbf{s}_{k,i}^{(l)})} \tag{6}$$

Then, information from all the keywords k_i bid by the advertiser a will be weighted aggregated to provide subspace-specific complementary:

$$\mathbf{s}_{a,t}^{(l+1)} = \frac{\mathbf{s}_{a,t}^{(l)} + \sum_{k_i} p_{a,k_i,t}^{(l)} \mathbf{s}_{k_i,t}^{(l)}}{||\mathbf{s}_{a,t}^{(l)} + \sum_{k_i} p_{a,k_i,t}^{(l)} \mathbf{s}_{k_i,t}^{(l)}||_2} \tag{7}$$

This single disentangled layer can be stacked to capture the high-order topology information. The outputs from the top layer are viewed as the final representations:

$$\mathbf{z}_{a,t} = \mathbf{s}_{a,t}^{(L)} \tag{8}$$

Finally, we use the dot product to measure whether an advertiser will bid a keyword:

$$y_{a,k} = \mathbf{h}_a^\top \mathbf{h}_k \tag{9}$$

The unsupervised training objective function should encourage nearby nodes to have similar representations, while enforcing that the representations of disparate nodes are highly distinct:

$$\mathcal{L}_{ae} = -\log(\sigma(\mathbf{h}_a^\top \mathbf{h}_k)) - \log(\sigma(-\mathbf{h}_a^\top \mathbf{h}_{\hat{k}})) \tag{10}$$

in which k is the keyword bid by the advertiser a and $\hat{k}$ is the negative samples which are topological far from a. The learned advertiser representations will be fed into the matching layer and updated by the relevance modeling loss.

3.5 Matching layer

Embeddings learned from above three towers are fed into the matching layer to get the final classification outputs. Here we implement the matching layer as a multi-layer perceptron (MLP) following previous works (Lu et al., 2020; Zhu et al., 2021; Li et al., 2016).

3.6 Objective Function

The output vector from the matching layer is denoted as $\mathbf{y}' \in \mathbb{R}^{1\times 2}$, which contains the predicted probabilities of the input tuple is relevant or not. Cross-entropy is selected as the loss function as follows:

$$\begin{aligned} \mathcal{L} &= \sum_{x\in\mathbf{L}} cross(\mathbf{y}, \mathbf{y}'), \\ cross(\mathbf{y}, \mathbf{y}') &= -\sum_i \mathbf{y}_i \log(\mathbf{y}'_i). \end{aligned} \tag{11}$$

4 Experiments

In this section, we extensively evaluate the proposed BGTR model over an industry dataset. In section 4.1, we present the statistics of the dataset and training details. Then we go through several SOTA baseline models in section 4.2. Section 4.3 exhibits the overall performance of BGTR and baseline models. Section 4.4 conducts two ablation studies to investigate the effectiveness of different GNN aggregation strategies and the disentangled advertiser encoder part. Finally, we study the performance sensitivities of BGTR on the neighbor sampling strategies and the neighbor numbers.

	Positive	Negative	All
Training	90,536	43,405	133,941
Validation	11,162	5,566	16,728
Test	10,366	4,928	15,294

Table 2: Statistics of the relevance modeling dataset.

4.1 Dataset and Training Details

The proposed BGTR model is extensively evaluated on a real-world industry dataset. Compared with the query-keyword pairs, it is more difficult to manually label the query-keyword-advertiser tuples as the annotators should be familiar with the background of advertisers. Thus, we adopt a two-stage annotating pipeline. In the first stage, each training sample will be labeled by 10 junior annotators. If the positive and negative scores are similar, the sample will be further labeled by 5 senior annotators. Finally, we achieve a dataset with 165,963 samples. As far as we know, this is the first triple-based dataset for relevance modeling, which is also much larger than the publicly available pair-based datasets (e.g., 32,000 for ESR[1] and 30,000 for MSLR [2]). As shown in Table 2, one can clearly see that this dataset is highly imbalanced. Thus, we select ROC-AUC score as the metric, which measures the size of area under the Receiver Operating Characteristic curve.

For the experimental settings, we use "Bert-base-uncased" in the huggingface[3] as the pre-trained BERT model. We save the checkpoints with best validation performance and then report their results on the test set. The number of training epochs is set to 10, and the size of minimal training batch is set to 64. Learning rate is set to 1e-5. In order to avoid the overfitting, we add the L2 regularization with the coefficient as 0.001. Model training is conducted on a Nvidia V100 GPU.

4.2 Baseline Models

We select several state-of-the-art methods as the baseline models in our experiments. These models can be divided into three categories: semantic-based models, naive GNNs and hybrid models.

Semantic-based models only capture semantic similarity inside the query-keyword pair without considering bidding graphs:

[1]https://data.world/crowdflower/ecommerce-search-relevance

[2]https://www.microsoft.com/en-us/research/project/mslr/

[3]https://github.com/huggingface/transformers/

Model	ROC-AUC
GAT	0.722
GraphSAGE-Mean	0.721
GraphSAGE-LSTM	0.727
C-DSSM	0.773
TwinBERT	0.796
TwinBERT + URL	0.795
TextGNN	0.797
BGTR	**0.801**

Table 3: Experimental results of different relevance models.

- **C-DSSM** (Shen et al., 2014a) is a latent semantic model that incorporates a convolutional-pooling structure over word sequences to learn representations for queries and keywords.

- **TwinBERT** (Lu et al., 2020) is a two-tower BERT-based model.

- **TwinBERT + URL** directly concatenates the URL of advertiser and the text of the input keyword, and then feeds the combined sentence into the keyword-tower.

For the adaptive keyword encoder, we introduce several popular GNN models to evaluate the effectiveness of the proposed tightly coupled framework. GNN models aggregate the pre-learned representations of co-order keywords as the final keyword embedding. We select the following two popular GNN models:

- **GraphSAGE** (Hamilton et al., 2017) aggregates the information over sampled neighbors and combines the aggregated information and center node's information together to generate the node representations.

- **GAT** (Veličković et al., 2017) introduces the multi-head attention mechanism to assign different neighbors with different weights in aggregation phase.

Hybrid models are capable of enjoying the merits from both semantic data and graph topology, in which BERT models and GNN models are jointly optimized under a loosely-coupled framework.

Aggregation strategy	with AE	w/o AE
LSTM	0.801	0.799
Mean-pooling	0.801	0.799
Self-attention	**0.802**	**0.800**

Table 4: Ablation studies on the aggregation strategy and advertiser encoder (AE). The presented performance metric is ROC-AUC.

- **TextGNN** (Zhu et al., 2021) fuses the text and graph information with a node-level aggregator. The keyword representation is first encoded by the BERT model, then combined together with the neighbor representations through a GAT model.

4.3 Experimental Results

Table 3 presents the ROC-AUC scores of the baseline models along with the proposed BGTR model. We repeat the training process three times and report the average ROC-AUC scores.

From the results, one can clearly see that the naive GNN models perform the worst. It may be due to the node textual features are pre-learned and fixed in the training phase, leading to the limited expression capacity. For the semantic-based two-tower models, TwinBERT outperforms C-DSSM by nearly 2%. This is reasonable as pre-trained models can provide a good starting point for the downstream tasks that leads to much better performance. It is worth noting that the performance of TwinBERT + URL slightly drops as the texts in the URLs are usually very obscure, which may introduce noises into the model training. TextGNN model outperforms semantic-based models, which verifies the effectiveness of the bidding graphs. Our proposed BGTR model outperforms the best baseline model (TextGNN) by more that 0.4% as it can effectively extract the valuable information of advertisers from the bidding graphs and tightly fuse the graph topology with the semantic texts.

4.4 Ablation Study

Here we perform the ablation study to measure the importance of different components in the proposed model. Specifically, we study the effectiveness of advertiser encoder and different aggregation strategies in the inter-node passing process. Table 4 presents the results of ablation studies.

Aggregation Strategy. In the inter-node passing component of advertiser-aware keyword en-

coder, we select the multi-head self-attention as the aggregation strategy to fuse the neighborhood. However, it is worthwhile to learn the performance of other types of aggregation strategies. Here we compare self-attention with mean-pooling and LSTM used in GraphSAGE (Hamilton et al., 2017). Results in Table 4 demonstrate that the proposed framework is quite stable to different aggregation strategies, in which self-aggregation method slightly outperforms others.

Advertiser Encoder. Here we aim to prove the effectiveness of advertiser encoder. As shown in Table 4, the performance of all models drop slightly without the disentangled advertiser encoder. This is due to the advertiser encoder can effectively fuse the bidding behaviors into the representations, leading to the better understanding of the advertiser-specific search intents.

4.5 Neighbor Sampling Analysis

Here we study the performance sensitivity of neighbor sampling from two aspects: sampling strategy and the number of neighbors. For the sampling strategy, we select the ANN (Approximate Nearest Neighbor) sampling and random sampling. ANN sampling samples the most similar co-order keywords based on their semantic closeness while random sampling simply randomly samples neighbors from co-order keyword set. The number of neighbor nodes is set to [2,4,6,8,10] to evaluate the model performance with different neighbors. Figure 4 presents the results. One can clearly see that with the increases of neighbor count, the performance keeps increasing. This is reasonable as more neighbors will bring abundant contextual information as complementary, yielding better model performance. ANN performs better than random sampling as ANN neighbors are more literately similar to the center keyword, while random neighbors may be unrelated keywords and may bring noises to the final keyword representation.

5 Related Work

In this section, we will briefly summarize the related works of relevance modeling in sponsored search. Traditional methods like LSA (Salakhutdinov and Hinton, 2009), LDA (Blei et al., 2012) and Bi-Lingual Topic Models (Gao et al., 2011) seek to mapping sentences to low-dimensional continuous vectors using shallow language representation models. Then the similarity can be calculated on this

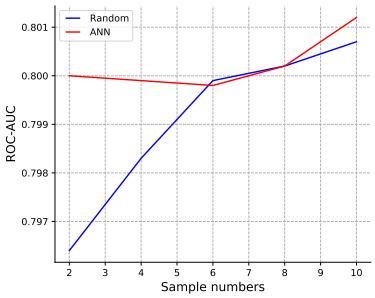

Figure 4: Neighbor sampling analysis

latent space. In recent years, with the success of deep learning in NLP area, deep semantic models, especially the siamese structure models is adopted in a range of works (Shen et al., 2014c,b; Hu et al., 2015; Tai et al., 2015; Gao et al., 2017; Li et al., 2017). Gao et al. (Gao et al., 2017) presents a deep semantic similarity model for recommending target documents to be of interest to a user based on a source document that she is reading with special convolutional-pooling structure. Some interaction-based structure (Wan et al., 2015; Yin et al., 2015; Yang et al., 2018) are also proven to be useful in relevance modeling. Yang et al. (Yang et al., 2018) propose an attention-based neural matching model with value-shared weighting scheme for combining different matching signals. Guo et.al. (Guo et al., 2016) employs a joint deep architecture at the query term level for relevance matching to bridge the gap between semantic matching and relevance matching. Mitra et al. (Mitra et al., 2017) propose a document ranking model composed of two separate deep networks that that matches the query and the document on separate representations. Bai et al. (Bai et al., 2018) propose query n-gram embedding to improve the modeling of query-ad relevance. Grbovic et al. (Grbovic and Cheng, 2018) propose a real-time personalization in search ranking and similar listing recommendations using listing and user embedding techniques. Huang et al. (Huang et al., 2020) design a unified embedding framework to model semantic embeddings for personalized search with various tricks including ANN parameter tuning and full-stack optimization.

6 Conclusion

In this paper, we thoroughly study the novel problem of advertiser-aware relevance modeling . The bidding behaviors of advertiser are incorporated to provide complementary information beyond the

semantic texts. We propose a triple-tower based model BGTR to deeply fuse the bidding graphs and the semantic information. Our proposal is extensively evaluated over an industry dataset, and the results demonstrate the superiority of the BGTR model.

References

Xiao Bai, Erik Ordentlich, Yuanyuan Zhang, Andy Feng, Adwait Ratnaparkhi, Reena Somvanshi, and Aldi Tjahjadi. 2018. Scalable query n-gram embedding for improving matching and relevance in sponsored search. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 52–61.

David M Blei, Andrew Y Ng, Michael I Jordan, and John Lafferty. 2012. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jianfeng Gao, Xiaodong He, and Li Deng. 2015. Deep learning for web search and natural language processing.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. 2017. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jianfeng Gao, Kristina Toutanova, and Wen Tau Yih. 2011. Clickthrough-based latent semantic models for web search. In *Proceeding of International Acm Sigir Conference on Research Development in Information Retrieval*.

Mihajlo Grbovic and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 311–320.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2015. Convolutional neural network architectures for matching natural language sentences.

Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2553–2561.

Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. Adsgnn: Behavior-graph augmented relevance modeling in sponsored search. *arXiv preprint arXiv:2104.12080*.

Chaozhuo Li, Senzhang Wang, Dejian Yang, Zhoujun Li, Yang Yang, Xiaoming Zhang, and Jianshe Zhou. 2017. Ppne: property preserving network embedding. In *International Conference on Database Systems for Advanced Applications*, pages 163–179. Springer.

Chaozhuo Li, Yu Wu, Wei Wu, Chen Xing, Zhoujun Li, and Ming Zhou. 2016. Detecting context dependent messages in a conversational environment. *arXiv preprint arXiv:1611.00483*.

Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model ensemble for click prediction in bing search ads. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 689–698.

Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. Twinbert: Distilling knowledge to twin-structured bert models for efficient retrieval. *arXiv preprint arXiv:2002.06275*.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014a. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 101–110.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014b. A latent semantic model with convolutional-pooling structure for information retrieval. pages 101–110.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014c. Learning semantic representations using convolutional neural networks for web search. In *International Conference on World Wide Web*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Computer ence*, 5(1):: 36.

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

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2015. A deep architecture for semantic matching with multiple positional sentence representations.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Guangzhong Sun, and Xing Xie. 2021. Graphformers: Gnn-nested language models for linked text representation. *arXiv preprint arXiv:2105.02605*.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. 2018. anmm: Ranking short answer texts with attention-based neural matching model.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Computer Science*.

Jiawei Zhang, Haopeng Zhang, Li Sun, and Congying Xia. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Liangjie Zhang, Tianqi Yan, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. *arXiv preprint arXiv:2101.06323*.