Heterogeneous Graph Neural Networks for Large-Scale Bid Keyword Matching

Zongtao Liu, Bin Ma, Quan Liu*, Jian Xu, Bo Zheng
Alibaba Group, China
{zongtao.lzt,mabin.mb,lq204691,xiyu.xj,bozheng}@alibaba-inc.com

ABSTRACT

Digital advertising is a critical part of many e-commerce platforms such as Taobao and Amazon. While in recent years a lot of attention has been drawn to *the consumer side* including canonical problems like ctr/cvr prediction, *the advertiser side*, which directly serves advertisers by providing them with marketing tools, is now playing a more and more important role. When speaking of sponsored search, bid keyword recommendation is the fundamental service. This paper addresses the problem of keyword matching, the primary step of keyword recommendation. Existing methods for keyword matching merely consider modeling relevance based on a single type of relation among ads and keywords, such as query clicks or text similarity, which neglects rich heterogeneous interactions hidden behind them. To fill this gap, the keyword matching problem faces several challenges including: 1) how to learn enriched and robust embeddings from complex interactions among various types of objects; 2) how to conduct high-quality matching for new ads that usually lack sufficient data.

To address these challenges, we develop a heterogeneous-graph-neural-network-based model for keyword matching named Het-Match, which has been deployed both online and offline at the core sponsored search platform of Alibaba Group. To extract enriched and robust embeddings among rich relations, we design a hierarchical structure to fuse and enhance the relevant neighborhood patterns both on the micro and the macro level. Moreover, by proposing a multi-view framework, the model is able to involve more positive samples for cold-start ads. Experimental results on a large-scale industrial dataset as well as online AB tests exhibit the effectiveness of HetMatch.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; **Computational advertising**; • **Computing methodologies** → *Neural networks*.

KEYWORDS

online advertisement, keyword recommendation, graph neural networks

*Corresponding author: Quan Liu, lq204691@alibaba-inc.com.
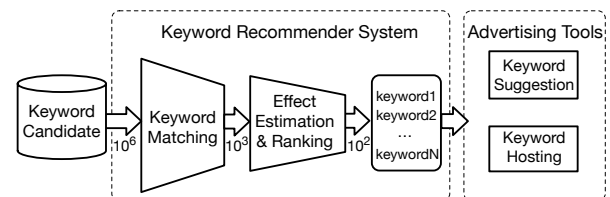
1 INTRODUCTION

Figure 1: A real-world integrated keyword recommender system.

Sponsored search is one of the main fashions of online advertising where advertisers acquire their desired ad impressions or clicks via bidding proper keywords. At the sponsored search platform of Alibaba, millions of advertisers in total manually add tens of millions of keywords every day, which reflects advertisers' strong bidding willingness to make their products obtain more potential consumers. Compared with such strong willingness, many advertisers lack expert knowledge to choose proper keywords thus fail to get expected ad impressions or clicks for their products. As illustrated in the previous empirical study [30], many advertisers tend to bid on a small number of popular keywords in the advertising platform, which makes those advertisers with low bid prices harder to obtain impressions. This challenge is also common at the sponsored search platforms in Alibaba, where only less than 10% of handcrafted selected keyword can obtain ad impressions on the next day. To improve marketing effectiveness and lighten the burden of advertisers, many sponsored search platforms are now offering different products that perform keyword recommendation, either in a white-box or a black-box fashion. White-box products offer many keyword candidates where advertisers can manually choose from, while black-box tools provide automatic keyword hosting services.

Like many other industrial recommender systems (RSs), an advertising keyword recommender system can be achieved via two phases: 1) keyword matching/retrieval which collects a subset of tokens among sea of words for a given ad; 2) efficient ranking which puts an order on individual subset based on relevance and estimated effect (C.f. Figure 1). In this work, we address the problem of keyword matching, which is a fundamental step of keyword

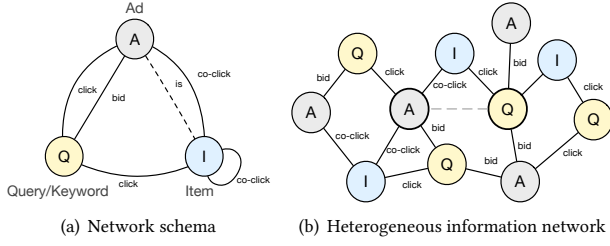(a) Network schema   (b) Heterogeneous information network

Figure 2: HIN schema for e-commerce bid keyword matching problem. An *ad* refers to an ad-group created in the advertising platform, a keyword can either denote a bidding keyword or an ordinary query searched by users, and an *item* denotes an ordinary goods in the e-commerce platform. Besides, each ad is corresponding to a unique item.

recommendation and also plays a determining role in the quality of the ranking stage.

Regarding the keyword matching or retrieval problem, various approaches have been explored over years, including text similarity, collaborative filtering, and topic-based ones [2, 13]. However, there are several limitations of existing methods: 1) these approaches neglect rich heterogeneous information hidden behind the simple <ad, keyword> pair (Figure 2 illustrates a toy example of HIN schema in which ads, keywords, and items are denoted by different types of nodes), which helps understanding the relevance between them ; 2) This also causes the cold-start problem to be worse for new ad units, especially those of new advertisers.

To utilize the rich information hidden behind ads and keywords, we resort to heterogeneous graph neural networks (HGNNs), which have various successful applications in modeling complex interactions between different types of objects [4, 6], and then integrate HGNNs in an embedding-based framework which is proven effective in multiple information retrieval tasks [19]. Embedding-based methods have been extensively studied in recommendation scenarios to improve the quality of matching. These methods aim to represent each node with different types of features in heterogeneous information network as a low-dimensional embedding vector, expecting that similar source objects (ads) and target objects (keywords) have similar embeddings. Among them, GNN-based approaches benefit from their strong ability to fuse relevant information from neighbors at different distances in the network, thus achieving state-of-the-art performance in matching task [26]. More recent studies have extended GNN to heterogeneous information networks to capture the complex interactions among various types of nodes[19]. Most of these studies achieve this goals by aggregating neighbors' information by sampling subgraphs via different metapaths [16, 32], where a metapath denotes a sequence of meta relations (C.f. Section 3).

For heterogeneous recommendation such as keyword recommendation, the selected metapaths vary a lot between different sides of a two-tower structure [16, 32], where a tower refer to a subnetwork to compute a node's final embedding, resulting that the sampled attributed subgraph in each tower is very different. More

specifically, the types of nodes, edges and their corresponding features in the same position of each subgraph vary a lot. In this way, it might be hard to guarantee that the final embeddings of two towers fall into adjacent feature space. Besides, it is also a challenging problem to filter irrelevant information when involving heterogeneous relations and attributes. This problem is especially severe in industrial web-scale scenes such as in Alibaba's e-commerce sponsored search platform. How to learn comprehensive and robust embeddings in modeling complex interaction among nodes is an intractable problem.

Another factor affecting the quality of keyword matching is the cold start of new ads. In Taobao, a leading e-commerce platform in China, millions of new ads are continuously created by advertisers every day. However, there are no user behaviors for these new ads throughout the platform. How to process these "cold start" ads is another challenging problem.

Motivated by the concerns mentioned above, this paper proposes a **Het**erogeneous graph neural network for keyword **Match**ing (HetMatch) for e-commerce sponsored search. To enhance the robustness of the extracted embeddings with various semenatics, on the macro level, we develop a Siamese network architecture by aggregating each tower's final embeddings and the averaged neighborhood embedding to conduct matching. In this way, the original heterogeneous matching problem between ads and keywords is transformed into a homogeneous matching between two <ad, keyword> pairs. On the micro level, we apply autoencoder in graph convolution as a fundamental component of our model to reduce the influence of noisy signals and the computational cost. Moreover, to improve the capability to match cold-start ads, we leverage a multi-view framework to use multiple types of relations besides click data of ads as our objectives.

Accordingly, our contributions are as follows:

- We propose a novel HGNN-based keyword matching framework that leverages the complex relational data behind ads and keywords. To extract enriched and robust embeddings among different relations, on the micro level, we apply an autoencoder in graph convolution to mitigate the irrelevant patterns in neighborhood; while on the macro level, we develop a Siamese neighbor matching layer on the top of HGNNs to involve more relevant neighborhood information.
- To introduce more supervised signals for alleviating the cold-start problem, we use a multi-view framework to learn the posterior probability of various objectives.
- Experiment results exhibit our model's advantages over other methods both on offline and online evaluation.
- To the best of our knowledge, it is the first work to apply a HGNN-based method to keyword recommendation, which consolidates the foundation of sponsored search.

2 RELATED WORK

Keyword Recommendation and Suggestion. Early methods of bid keyword recommendation in sponsored search mainly focus on retrieving keywords from the perspective of relevance. These methods can be broadly categorized into the following types: collaborative methods [2], proximity-based methods [1, 13], and topic-based methods [3]. Some collaborative methods like [2] aim to mine

phrases that co-occur with the seed queries, which is applied in Google's Adword Tools. Proximity-based approaches mainly focus on designing different distance metrics to evaluate the similarity between queries, which can be further divided into network-based methods [13] and kernel-based methods [1]. Topic-based methods use a hierarchical or unsupervised approach to cluster queries to topic [3]. These methods mainly address the relevance of retrieved corpus and ignore modeling the effect of selected keywords; thus may not perform well in practice.

Another line of keyword recommendation focuses on retrieving the keywords from click-logs and estimate the user-click or impression brought by selected keywords [7, 15, 24]. For instance, Fuxman et al. [7] propose algorithm within Markov Random Field model to estimate user clicks. However, these methods are computationally expensive and lack the generalization ability for new ads; thus are usually applied to ranking retrieved keywords rather than performing matching tasks.

Representation Learning on Heterogeneous Networks. Network representation learning (NRL) aims to automatically encode node information and network structure to fixed-sized latent representation, which can be used in downstream graph mining tasks. Over the years, quite a few NRL methods have been proposed. These methods can be divided into random-walk-based methods [4], factorization-based methods [17] and GNN-based methods [18, 25, 29]. Earlier works mostly focused on studying the network representation learning method on homogeneous graphs. Recently, inspired by the past homogeneous NRL, studies have attempted to extend homogeneous NRL methods on heterogeneous networks. And the core problem is to fuse diverse node attributes and different types of relations into a latent vector; thus, the extracted embeddings can preserve both semantic and structural properties among nodes. Based on previous random-walk-based homogeneous NRL, Dong et al. [4] and Fu et al. [6] introduce metapath-based random walk strategies, in which the walker is confined to transit based on given metapaths. On the other line of works, applying GNNs on heterogeneous networks has also been proven a promising direction [18, 25, 29]. For instance, Wang et al. [25] maintains different weights for different metapath-defined edges when applying aggregation in the graph attention networks. However, current HGNN-based methods do not explicitly address the importance of filtering irrelevant patterns propagated through complex interactions, which is achieved by our extended version of graph convolution.

Heterogeneous-Information-Network based Recommendation. More recently, leveraging heterogeneous information networks (HIN) becomes an emerging direction in recommender systems due to its capability of characterizing complex objects and rich relations [19]. For instance, Feng and Wang [5] proposed to alleviate the cold start issue with heterogeneous information network contained in the social tagged system. Metapath-based methods were introduced into hybrid recommender system in [28]. Recently, Hu et al. [9] leveraged metapath-based context in top-N recommendation. However, none of them addresses the problem of the heterogeneity of metapaths between two towers, which might lead to difficulties for learning a robust embedding for matching with very different computational graphs.

3 PROBLEM SETUP

Recently, several researches have explored how to utilize the rich relations and complex objects of heterogeneous information network (HIN) in recommendation systems, which is an auspicious direction. Keyword recommendation also deals with various types of objects and rich relations (C.f. Figure 2), thus it would be promising to leverage the HIN-based recommendation paradigm. Here we first give the necessary concepts about HIN.

Definition 3.1. **Heterogeneous Information Network** [20]. A heterogeneous information network (HIN) can be denoted as $G = (\mathcal{V}, \mathcal{E}, \phi, \psi)$, consisting of an object set $\mathcal{V}$ and a relation set $\mathcal{E}$. Each node v owns a type $\phi(v)$, and each edge e has a type $\psi(e)$. $\mathcal{T}$ and $\mathcal{R}$ denote the sets of predefined object types and relation types, where $|\mathcal{T}| + |\mathcal{R}| > 2$. $\mathcal{V}$ can be written as $\mathcal{V} = \mathcal{V}_{t_1} \cup ... \cup \mathcal{V}_{t_i} \cup ... \cup \mathcal{V}_{t_{|T|}}$, where $t_i \in \mathcal{T}$ and $\mathcal{V}_{t_i}$ represents the node set with type t_i. Besides, each node $v_t \in \mathcal{V}_t$ is augmented with node attribute $\mathbf{x}^t \in \mathbb{R}^{d_t}$, where d_t is the attribute dimension of the nodes typed t. Similarly, $\mathcal{E}$ can be written as $\mathcal{E}_{r_1} \cup ... \cup \mathcal{E}_{r_i} \cup ... \cup \mathcal{E}_{r_{|\mathcal{R}|}}$, where $r_i \in \mathcal{R}$ and $\mathcal{E}_{r_i}$ represents the edge set with type r_i.

More specifically, in the scenario of keyword recommendation, we denote the whole node set $\mathcal{V} = \mathcal{V}_A \cup \mathcal{V}_Q \cup \mathcal{V}_I$, where $\mathcal{V}_A$, $\mathcal{V}_I$ and $\mathcal{V}_Q$ represents the ad set, keyword set, and item set, respectively. More specifically, an *ad* refers to an ad-group created in the advertising platform, a keyword can either denote a bidding keyword or an ordinary query searched by users, and an *item* denotes an ordinary goods in the e-commerce platform. Among these nodes, there exists various types of relations which capture different semantic connections, such as the bid or click relations between ads and keywords. In order to capture the semantic and structural relation between different objects, the metapath is proposed by [21] as a relation sequence connecting two objects, and is widely applied in HIN model research.

Definition 3.2. **Metapath and Metapath-guided Neighbors** [21]. For a relation $e = (s, t)$ linking from node s to t, its meta relation is defined as $< \phi(s), \psi(e), \phi(t) >$. A meta-path p is then defined as a sequence of meta relations $t_1 \xrightarrow{r_1} t_2 \xrightarrow{r_2} \cdots \xrightarrow{r_l} t_{l+1}$, which describes a composite relation $R = r_1 \circ r_2 \cdots \circ \cdots r_l$ between nodes a_1 and a_{l+1}, where $\circ$ denotes the composition operator on relations. Furthermore, the metapath-guided neighborhood is defined as the set of all visited nodes when a given node v walks along the given metapath p.

Finally we present the problem definition of HIN-based keyword matching.

Definition 3.3. **HIN-based Ad-Keyword Matching Problem.** Given a HIN $G = (\mathcal{V}, \mathcal{E})$, let $\mathcal{A} = \{a_1, a_2, ..., a_N\} \subset \mathcal{V}_\mathcal{A}$ be a set of selected ads to match keywords, and $\mathcal{E}^c, \mathcal{E}^t \subset \mathcal{A} \times \mathcal{V}_Q$ be the candidate relation set and target relation set respective, where '$\times$' represents the Cartesian product operator between two set. More specifically, each ad a_i corresponds to a target keyword set $Q_i^t \subset \mathcal{V}_Q$ and a candidate keyword set $Q_i^c \subset \mathcal{V}_Q$.

The ad-keyword matching problem can be formulated as a recall optimization task [11]. Given a set of ads $\mathcal{A} = \{a_i\}_{i=1}^{N}$ with their corresponding candidate relation sets $\mathcal{E}^c = \bigcup_{i=1}^{N}(a_i \times Q_i^c)$ and the target relation sets $\mathcal{E}^t = \bigcup_{i=1}^{N}(a_i \times Q_i^t)$, the problem is to pick a retrieved relation set $\mathcal{E}^o = \bigcup_{i=1}^{N}(a_i \times O_i^K)$, where O_i^K represents the retrieved keywords for ad a_i sized no more than K, the object is to maximize the total recall ratio:

$$Recall@K = \frac{\sum_i^N |O_i^K \cap Q_i^t|}{\sum_i^N |Q_i^t|} \tag{1}$$

Usually, the candidate set of a matching problem is set as a group of target nodes which is related to the source nodes under a certain criterion. In our task, we define the candidate set Q_i^c of an ad a_i as the keywords that share the same category with a_i, which is adjusted by a trained category classifier. The target relation set $\mathcal{E}^t$ contains the node pair of <ad, keyword> which would bring user clicks in the future. In other words, we aim to optimize our matching model in that the matched keywords can cover as many effective bidding keywords as possible.

4 METHOD

In recent years, attention in the field of RS is increasingly shifting towards HIN-based recommendation. We extend this paradigm of RS for bid keyword matching problem and design a hierarchical network architecture to model the enriched interactions behind ads and keywords in a robust way. On the micro level, we leverage metapath-based graph convolution to aggregate neighborhood context from different perspectives and utilize an autoencoder module to filter irrelevant neighborhood patterns. On the macro level, we design a Siamese network structure to enhance the complementary patterns in the neighborhood. To further alleviate the cold start problem of newly created ads, we extend our framework as a multi-view matching problem among different relation targets between ads and keywords. In this way, more supervising information are introduced and the associated tasks can improve each other's performance by sharing information. In this section, we describe the details of our proposed model, **Het**erogeneous Graph Neural Network for keyword **Match**ing (HetMatch). Figure 3 shows the architecture of our proposed model.

4.1 Node-Level Fusion

Before integrating data from metapath neighbors for each node, we first extract heterogeneous feature vector $\mathbf{x}^t \in \mathbb{R}^{d_t}$ from $\mathcal{X}^t$ for each node $v_t \in \mathcal{V}_t$ and transform it as a fixed-length embedding. We list the details of features in Table 1. These features can be categorized into two types, i.e., id features and numeric features. We transform each numeric feature as a discrete value, which represents the quantile of its feature distribution. After discretization, we can acquire the embedding of each feature via its corresponding look-up table. To lighten the computation burden, the same features from different node types will share one look-up table. For instance, the feature *terms of title* appears both in ad's and item's feature list, and thus the embedding of this feature shall be acquired from the same look-up table. We then concatenate these embeddings and feed the concatenation into a type-specific neural network f_t to get the node-level embedding $\mathbf{h}_{v_t}$.

4.2 Subgraph-Level Fusion

Metapath. After fusing the node-level information, the next step is to involve the rich context information around each node. Under the paradigm of metapath-based heterogeneous GNNs, multiple metapaths are sampled which play different roles in capturing the structural and semantic context. We thus introduce the metapaths used in our model, which can be categorized into the following two groups:

(1) *Bid-based group*: The subgraphs based on bid relations around a given ad or keyword can directly characterize the bidding environment around it, which involves the competitive ads that bid on the shared keywords and a surge of keywords that the competitive ads are interested in. We use the following four metapaths constructed by user clicks and advertiser bids to capture such environment:

- $q \xrightarrow{click} a \xrightarrow{click} q$, $q \xrightarrow{click} a \xrightarrow{bid} q$
- $a \xrightarrow{click} q \xrightarrow{click} a$, $a \xrightarrow{bid} q \xrightarrow{click} a$

where the user-click relations can stand for the bids that can directly bring clicks, and the ordinary bid relations can make appropriate supplements to the cold-start ads.

(2) *Item-based group*: Sometimes a user might pay attention to an ad and an ordinary item in the same page view. These items can build bridges between ads and related keywords that are not directed connected and provide extra useful semantics that helps capture richer context information. Besides, those co-clicked items can also offer similar textual and behavioral patterns that can characterize how the connected nodes are like.

- $q \xrightarrow{click} i \xrightarrow{co-click} a$
- $a \xrightarrow{co-click} i \xrightarrow{click} q$

Graph Convolution Layer with AutoEncoder. After introducing the metapaths used in our model, we then present how we aggregate the node-level embeddings robustly and efficiently given a certain metapath. The core idea of graph neural networks is to iteratively aggregate feature information in the neighborhood by a local filter. However, the neighborhood information aggregated by different heterogeneous relations might contain irrelevant information. Without loss of generality, we apply the autoencoder, which is successfully used in previous tasks for learning the compressed representation for denoising [23], to filter irrelevant neighborhood patterns and meanwhile improve the efficiency of aggregating. The

Table 1: List of features in this paper.

| Node Type	Features
ad	ad id, terms of title/description, category path, brand, properties and shop information
item	item id, terms of title/description, category path, brand, properties and shop information
keyword	keyword id, terms of keyword, predicted category, average bid price of the keyword, the number of bid on the keyword, the total in-shop count led by the keyword
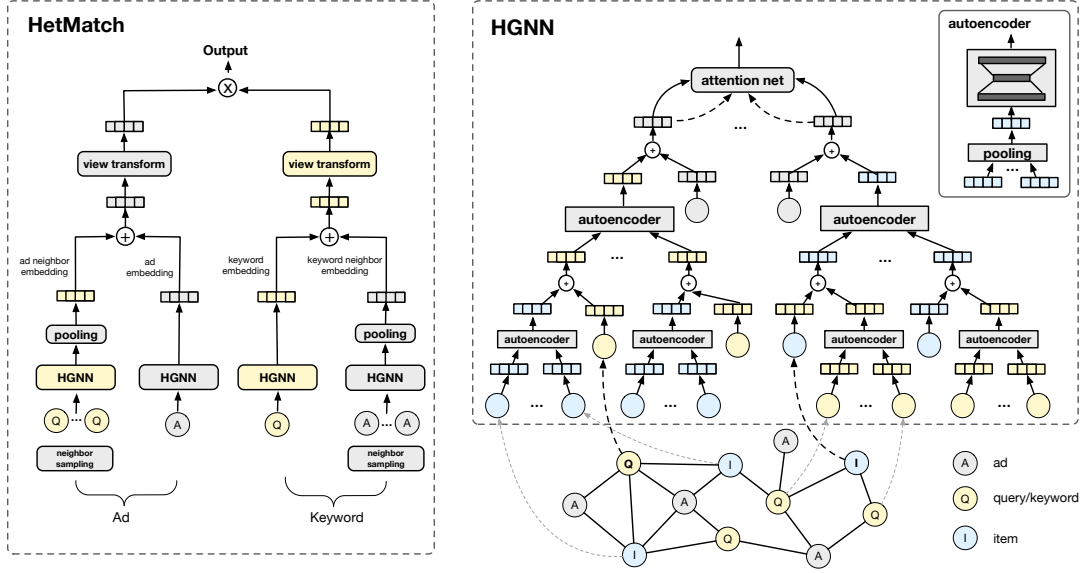
Figure 3: Architecture of our proposed model. The left part is the two-tower structure of our proposed method in which we use a Siamese neighborhood matching layer to involve more homogeneous node patterns and utilize a view-transformation module to involve more supervised signals. The right part is the metapath-based heterogeneous GNN in which we use an autoencoder to reduce the influence of noisy signals and the computational cost.

process of graph convolution can be formulated as below:

$$h_{\mathcal{N}_r(v)}^{k-1} = \textbf{AGGREGATE}(h_u^{k-1}, \forall u \in \mathcal{N}_r(v)) \quad (2)$$

$$h_v^k = \sigma(f(h_v^{k-1}) + g(h_{\mathcal{N}_r(v)}^{k-1})) \quad (3)$$

$$f(h_v^{k-1}) = \mathbf{W}^k \cdot h_v^{k-1} \quad (4)$$

$$g(h_{\mathcal{N}_r(v)}^{k-1}) = \mathbf{U}^k \cdot \sigma(\mathbf{V}^k \cdot h_{\mathcal{N}_r(v)}^{k-1}) \quad (5)$$

where h_v^k denotes the hidden state of node v after the k-th convolution layer, and **AGGREGATE** is a pooling operator (*mean*, *sum*, and etc.) and $h_{\mathcal{N}_r(v)}^{k-1}$ is the neighborhood vector of v that incorporates the surrounding information of v given a meta relation r. Without loss of generality, we instantiate the pooling function as a *sum* operator. In (3), $f(\cdot)$ and $g(\cdot)$ are two distinctive data-driven functions that maps h_v^{k-1} and $h_{\mathcal{N}_r(v)}^{k-1}$ to a new feature space respectively. Here $\mathcal{N}_r(v)$ contains the neighbors of v with the top-m largest edge weights. The motivation we select two distinctive functions is due to the heterogeneity of these two hidden representations which correspond to different node types. For the hidden features of the node itself in (4), we directly use a linear projection with a weight matrix $\mathbf{W}^k \in \mathbb{R}^{d \times d}$. For the neighborhood vector in (5), we first compress it to a low-dimension vector sized l by a linear transformation with $\mathbf{V}^k \in \mathbb{R}^{d \times l}$, where $l < d$, followed by a activation function σ. For term convenience, we name l as latent feature size. We then apply another linear combination with $\mathbf{U}^k \in \mathbb{R}^{l \times d}$ to map the intermediate hidden features to the original dimension sized d. We select such autoencoder-based method for the neighborhood vector because it can be viewed as a noise filter from the neighborhood context which usually contains some irrelevant information. Besides, it can also reduce the computational complexity compared with the direct assignment of a weight matrix sized $d \times d$ on the hidden features

from $O(d^2)$ to $O(d \cdot l)$. Although the graph attention network (GAT) [22, 27] also addresses the problem to extract the most influential information from neighbors by the attention mechanism, it has been experimentally verified that GAT performed the same as or worse than GCN in noisy graphs [33]. This is because GAT introduces too many parameters when learning attention coefficients, which leads to overfitting to noise. Besides, compared with our design, GAT is more computation expensive.

After feeding the node information to K such layers, which are defined by metapath p, we can eventually obtain the node semantic embedding h_v^p. To reduce the computational cost, the network parameters of the graph convolution are shared across different nodes at the same layer.

Semantic Attention Layer. We next introduce the semantic attention layer, which aims to fuse multiple embeddings extracted based on various metapaths. The general idea is as follows: different metapaths reveal different aspects of node context; thus, there is a necessity to aggregate the semantics revealed from different metapaths. To address this issue, we follow the idea proposed in [?], which uses a self-attention mechanism to capture the diverse semantics revealed from different metapaths. The mathematical expressions of the semantic fusion are as follows:

$$\tilde{h}_v = \sum_{p \in P} w_p \cdot h_v^p \quad and \quad w_p = \frac{exp(W_{att} \cdot h_v^p)}{\sum_{p \in P} exp(W_{att} \cdot h_v^p)} \quad (6)$$

where w_p is the learned importance weight of metapath p, which is computed by a scaled self-attention mechanism from (6), and $W_{att} \in \mathbb{R}^{d \times 1}$ is the weight matrix for mapping the original hidden representation h_v^p to a scalar which will be scaled to the importance weight of the corresponding metapath p.

4.3 Siamese Neighbor Matching

In the past two sections, we have presented the methodology to fuse node-level and subgraph-level information. In an ordinary matching model, the next phase is usually to feed the source and target embeddings into a contrastive loss function, like in [12]. Before calculating the loss function, in this section, we design a Siamese neighbor matching layer to transform the original matching problem between heterogeneous nodes to a matching problem between two pairs of <ad, keyword>, which is illustrated in the left part of Figure 3. Each pair contains the source or target node and its most influential neighbors. More specifically, the <ad, keyword> pair in the tower of ads contains the embedding of source ad v_a and the averaged embedding of v_a's κ most influential neighbors with type of *keyword*, while the <ad, keyword> pair in the tower of keywords contains the embedding of target keyword v_q and the averaged embedding of its κ most influential neighbors with type of *ad*. We finally combine the averaged semantic embedding of a node's most influential neighbors with the node itself.

The motivation here is that although in a heterogeneous matching problem, the embeddings of both the source node (ad) and the target node (keyword) are expected to be close in feature space by optimization, there is no explicit guarantee for such expectations due to their differences in the network structures, parameters, and metapaths to compute the final embeddings. Besides, by introducing so much side-information brought by the heterogeneous graph neural networks, it would become more challenging to satisfy such assumption. In contrast, compared with only utilizing the source semantic embedding itself, introducing the averaged semantic embedding of its influential neighbors (named neighborhood embedding) would help match with target embedding. This is because they share the same raw feature types, network structures, metapaths, and parameters. Furthermore, the influential neighbors can also directly characterize what the source nodes are like in an e-commerce scenario. The mathematical expressions are formulated as below:

$$z_{v_a} = \tilde{h}_{v_a} + \mathbb{E}_{u_q \in \mathcal{N}_\kappa(v_a)} \tilde{h}_{u_q} \quad and \quad z_{v_q} = \tilde{h}_{v_q} + \mathbb{E}_{u_a \in \mathcal{N}_\kappa(v_q)} \tilde{h}_{u_a} \quad (7)$$

where $\mathcal{N}_\kappa(v_a)$ denotes κ most influential keywords bid by node v_a; similarly, $\mathcal{N}_\kappa(v_q)$ denotes κ most influential ads bidding on keyword v_q. In this way, the whole structures to compute z_{v_a} and z_{v_q} are symmetric, which is also the reason we name this network Siamese neighbor matching layer. It's important to note that Siamese neighborhood matching mechanism differs from the dual matching scheme used in [14, 16, 31]. These studies design symmetric network structures for heterogeneous objects in two-tower; however, their neighborhood embedding is computed based on different metapaths and network parameters thus tend to be hard for heterogeneous matching.

4.4 Multi-view Objectives and View Transformation

For improving the effectiveness of matching with cold-start ads, we introduce different types of relations between ads and keywords as our objectives. More specifically, we select click relations between ads and keywords, bid relations between ads and keywords, and click relations between keywords and items (each ad corresponds to

an item (Figure 2)) as our objectives. As different type of objectives might have different feature space to match, we use a view transformation function $f_{trans}(\cdot)$ to map the original embeddings z_v to a view-specific embedding z_v^r, where $f_{trans}(\cdot)$ is a view-specific multi-layer perceptron and r represent the view type.

4.5 Model Learning

We train our HetMatch in a supervised manner using a contrastive loss used in [12]. The basic idea is that we aim to maximize the inner product of positive pairs, i.e., the transformed embedding of the source ad and the corresponding related keyword. Meanwhile, we also want the inner product of negative examples, the source ad and its unrelated keyword, to be smaller than that of the positive sample. To achieve this, we compute the posterior probability of a keyword given an ad from the semantic relevance score between them through a softmax function:

$$p(v_q|v_a, Q') = \frac{exp(\gamma \cdot z_{v_a} \cdot z_{v_q})}{\sum_{v_{q'} \in Q'} exp(\gamma \cdot z_{v_a} \cdot z_{v_{q'}})} \quad (8)$$

where γ is a smoothing factor in the softmax function. Q' denotes a subset of candidate keywords to match. For each positive pair, denoted by (v_a, v_{q+}) where v_{q+} is the relevant keyword, Q' includes v_q^+ and five randomly selected keywords in the candidate set $Q_{v_a}^c$. The candidate set $Q_{v_a}^c$ contains the keywords that are discriminated by a category classifier as the same category as ad v_a.

In the training phase, we optimize the whole networks' parameter set by maximizing the likelihood of the posterior probability of all keywords across the training set. Equivalently, we acquire to optimize the following loss function:

$$L(\Theta) = - \sum_{(v_a, v_q^+, Q')} p(v_q^+|v_a, Q') \quad (9)$$

where Θ represents the parameter set of the whole networks which is updated iteratively by an Adam optimizer.

4.6 Deployment

Pipeline of Keyword Recommendation. Keyword recommendation is a multi-stage system, consisting of matching, filtering and ranking phases. In the keyword matching phase, we first handle network construction from user behavioral records and advertiser bidding records offline by MapReduce downstream in the Alibaba Cloud platform[1]. During the training and inference, we leverage the distributed deep learning framework XDL[2] and graph search engine, Euler[3], to conduct run-time subgraph extraction and parameter update in a distributed fashion. After inference, ad embeddings and keyword embeddings are stored in a database for online services, and we use an approximate nearest neighbor (ANN) search engine to retrieve the most relevant keywords for each ad. In the filtering phase, we use a term-match based relevance model to filter the unrelated keywords for each ad. Lastly, in the ranking phase, we use an MLP-based model with enriched features to estimate the potential number of clicks brought by each remained keyword and rank those keywords based on the estimated values. In this way,

---

[1]https://us.alibabacloud.com
[2]https://github.com/alibaba/x-deeplearning
[3]https://github.com/alibaba/euler

we can recommend advertisers with the keywords that can bring as many user clicks as possible.

Acceleration. HetMatch can be implemented by sampling sub-graphs separately for each node and computing each node's corresponding final embedding via neural networks, which is computation consuming. As the extracted subgraphs based on the top-k sampling strategy of a particular node and its influential neighbors usually share plenty of relation-paths, we lighten the computation burden by only calculating the intermediate embedding once for each distinctive relation-path.

Negative Sampling. As illustrated in Section 4.5, for each positive pairs, we need to sample multiple negative keyword nodes for training. The whole procedure for negative sampling can be divided into two phases: 1) we calculate the weights (square root of the searched count) of keywords in each leaf category offline. 2) At run time of training, for each positive <ad, keyword>, we pick five negative keywords in the same leaf category by weighted random selection with the help of the graph index engine Euler.

5 EXPERIMENTS

5.1 Experimental Setup

Dataset Description. We evaluate our proposed method based on a real-world dataset collected from the search logs in Taobao platform and the advertiser behavioral records in Alibaba's e-commerce sponsored search platform. The dataset covers seven consecutive days of these records, spanning from August 19th, 2020 to August 26th, 2020. We construct our heterogeneous network from these logs based on the network schema illustrated in Figure 2, where the edge weight (the importance of an edge) is defined as the appearing times of a particular relation in the records. For the training set, we sample 10M ad-keyword pairs in each view. The target relation set contains 50M click relations between ads and keywords in the constructed network. To prevent information leakage, we drop the relations in the target relation set from the original network. The more specific data statistical information is exhibited in Table 2.

Table 2: Statistics of the Dataset.

Relation(A-B)	#A	#B	#A-B	#labeled A-B	#target A-B
a_*click*_q	50M	10M	500M	10M	50M
a_*bid*_q	50M	10M	5B	10M	-
i_*click*_q	100M	100M	10B	10M	-
a_*coclick*_i	5M	50M	500M	-	-

Baselines. To demonstrate our proposed method's effectiveness, we compare our method with multiple baseline methods and their variants. To ensure the fairness of the comparison, the results we reported are all under the multi-view learning framework mentioned in Section 4.4 unless otherwise stated.

- **Term-Match**: This is a keyword retrieval method that extracts related keywords by calculating the term similarity between ads' titles and keywords, which is one of the keyword matching models currently deployed in Alibaba's e-commerce sponsored search platform.

- **DSSM**: This is a two-tower matching model that projects the features of objects to a low dimensional space via multi-layer perceptrons (MLPs) [12].
- **HAN**: We replace the MLPs in DSSM by heterogeneous attention networks (HANs) proposed in [25]. The metapaths to aggregate neighbors are mentioned in Section 4.2.
- **IntentGC**: This is a dual-HGCN-based recommendation model that captures heterogeneous relations between nodes with the same node types [31].
- **HetMatch**: This is our proposed model. We set the embedding length d as 64 and the latent size l as 16. The model is optimized using an Adam optimizer with a learning rate of 0.03, and the batch size is 512.
- **DSSM(s), HAN(s), IntentGC(s)**: Based on DSSM, HAN or IntentGC, we add the Siamese matching layer between the output of multi-layer perceptrons and the multi-view transformation layer's input.
- **HetMatch $_{\backslash s}$**: This model drops the Siamese neighbor matching layer in HetMatch. The remaining settings for this variant are the same as for our proposed methods.
- **HetMatch $_{\backslash v}$**: This model only utilizes click data between ads and keywords as the labeled positive samples.
- **HetMatch $_{\backslash a}$**: This variant replaces the graph convolution layer with autoencoder of HetMatch by GraphSage [8].
- **HetMatch $_{bid}$, HetMatch $_{item}$**: Each of these variants only use the bid/item relations to aggregate neighbors' information.

Table 3: Performance comparison.

K Method	100	200	500	1000
		Recall@3K		
Term-Match	10.74%	12.79%	16.10%	16.74%
DSSM	13.73%	23.15%	41.34%	57.74%
HAN	16.72%	27.65%	47.91%	64.83%
IntentGC	16.97%	26.63%	43.57%	57.04%
DSSM(s)	16.45%	27.24%	47.32%	64.35%
HAN(s)	17.52%	28.82%	49.75%	**67.33%**
IntentGC(s)	17.73%	27.93%	45.68%	60.01%
HetMatch $_{\backslash a}$	18.82%	29.60%	48.34%	63.31%
HetMatch $_{\backslash s}$	19.32%	30.28%	49.77%	65.63%
HetMatch $_{\backslash v}$	17.99%	23.91%	26.66%	47.43%
HetMatch	**19.82%**	**30.93%**	**50.49%**	67.16%
HetMatch $_{bid}$	15.97%	26.74%	46.33%	64.81%
HetMatch $_{item}$	15.78%	26.62%	47.48%	66.11%

5.2 Performance Evaluation

As our task focuses on retrieving hundreds of keyword candidates, we use the recall rate as our evaluation metrics as illustrated in Section 3, which is also adopted in [11]. As our model is under a multi-view framework (the number of view is 3), we use Recall@3K instead of Recall@K as our metrics. For the approaches under the multi-view framework, we retrieve each ad's the top-K most related keywords in each view, and aggregate these relations as the final retrieved set O^{3K} to compute Recall@3K. To ensure fair comparison, for those methods that are not under a multi-view framework (Term-Match and HetMatch $_{\backslash v}$), we directly retrieve the top-3K most related keywords for each ad to compute Recall@3K.

Table 4: Performance comparison in different views.

View	Recall@K (ad-click)				Recall@K (item-click)				Recall@K (ad-bid)			
K	100	200	500	1000	100	200	500	1000	100	200	500	1000
DSSM	4.52%	7.71%	14.46%	22.40%	4.63%	7.87%	14.61%	22.26%	4.46%	7.68%	14.18%	22.03%
HAN	5.83%	9.90%	18.38%	27.47%	5.78%	9.80%	18.30%	27.44%	5.80%	9.81%	18.06%	26.69%
IntentGC	6.33%	10.23%	18.40%	27.42%	6.16%	10.02%	18.19%	27.34%	6.13%	10.09%	18.34%	27.31%
HetMatch $_{\setminus v}$	4.51%	7.67%	14.72%	24.39%	-	-	-	-	-	-	-	-
HetMatch	**7.07%**	**11.21%**	**19.75%**	**28.99%**	**7.00%**	**11.14%**	**19.70%**	**28.96%**	**6.45%**	**10.52%**	**18.84%**	**27.82%**
HetMatch $_{bid}$	5.58%	9.50%	18.00%	27.55%	5.47%	9.34%	17.74%	27.33%	5.49%	9.41%	17.70%	26.68%
HetMatch $_{item}$	5.53%	9.41%	17.71%	26.77%	5.61%	9.57%	17.98%	27.10%	5.44%	9.31%	17.52%	26.30%

Table 3 reports the performance of our proposed method compared to other competing methods or some variants from them. It can be seen from the table that our model can consistently outperform all comparative approaches or variants by achieving the highest recall rate in most cases. Firstly, we compare the relevance-based model deployed in our platform with other embedding-based retrieval methods. We observe that our relevance-based model (Term-Match) cannot achieve a good performance, especially when K is large. This finding demonstrates that the retrieval model that only relies on text relevance might not have good practical performance. Then we compare HetMatch with DSSM that does not utilize neighbor information. We note that our method achieves much better performance, as HetMatch can capture auxiliary information from heterogeneous relationships. Besides, we compare HetMatch with GNN-based methods (HAN, IntentGC); here, HAN and IntentGC also performs the aggregation operations on neighbors, but it does not explicitly consider the misleading influence brought by noisy links and features, and the attention mechanism used in HAN's aggregator might lead to overfitting.

To further demonstrate how our Siamese neighbor matching layer helps matching task, we conduct several ablation studies on DSSM(s), HAN(s), IntentGC(s) and HetMatch $_{\setminus s}$. The comparison between DSSM with DSSM(s), HAN with HAN(s), IntentGC with IntentGC(s) and HetMatch with HetMatch $_{\setminus s}$ presents the effectiveness of introducing Siamese neighbor matching in a DNN-based matching task. Interestingly, we observed that adding our Siamese neighbor matching layer to a simple DNN-based matching approach (DSSM) can significantly improve performance, achieving very close performance to other GNN-based methods. This again shows the importance of conducting Siamese neighbor matching. We can also note that HetMatch surpasses the performance of HetMatch $_{\setminus a}$. One reason for this is that the autoencoder based graph convolution layer prevents misleading information from propagating from neighbors and thus enhances the quality of neighbor embeddings. Besides, by eliminating the multi-view framework, we observe a drastic drop in performance, which demonstrates the need to involve different types of objectives in learning.

We also explore how our proposed model performs with different groups of metapaths. Compared with sampling neighbors based on both bid-based and item-based metapaths, sampling neighbors with only one group of metapath harms the performance. This is because the bid-based and item-based relations can provide useful information from different aspects, which can complement each other. It would also be interesting to note that the performance of HetMatch $_{bid}$ is better than the performance of HetMatch $_{item}$ when K is no

more than 200, and vice versa when K is more than 200. This is because the item-based metapaths brings more diverse neighbors compared with those sampled based on the bid-based metapaths (since this group is not directly related to the object task), which might hamper the quality of matched elements whose matching score is relatively high, but improve the total recall rate when K is relatively large. This also shows the importance of selecting proper types of metapaths in the task of keyword matching.

5.3 Performance Evaluation in Different Views

This section aims to compare the performance of different approaches for each view objectives (Table 4). Our model achieves state-of-the-art performance in each view, and its improvement on the performance over other baseline methods is significant. This finding demonstrates our model can achieve a good performance not only in total but also in each channel. Another interesting finding is that we note that the performance in the channel of bidding relations is worse than that in the channels of item-clicking and ad-clicking in most cases. This is because the labels of bidding relations are extracted by advertisers' manual selection of keywords, which might introduce some non-professional bidding behaviors and achieve performance. Besides, we also note that HetMatch achieves better performance over HetMatch $_{\setminus v}$ in the view of ad-click, which demonstrates that learning with auxiliary labels in other views can improve the performance in the each view.

5.4 Performance Evaluation on Cold Start Ads

Conducting cold-start recommendation is a critical problem for a recommender system, which is not an exception for keyword recommendation. In this section, we present the experimental results for the cold-start ads. More specifically, Table 5 reports the performance of different methods on the ads which are newly created on August 26, 2020. Unlike the target set we used in Section 5.2, we select the candidate set of a new ad as the keywords that bring user clicks to the ad in the next 14 days. The performance of these methods is consistent with that in Section 5.2. HetMatch performs the best across all the competitive approaches, which demonstrates the HetMatch works well on the whole dataset and has an excellent capability to retrieve useful keywords for new ads. More specifically, we find that learning with various groups of metapaths and using autoencoder module in graph convolution and Siamese neighbor matching layer can improve the performance on cold-start ads.

Table 5: Performance comparison of cold-start ads.

K Method	100	200	500	1000
		Recall@3K		
Term-Match	15.14%	17.10%	18.82%	19.86%
DSSM	13.02%	21.90%	38.84%	53.82%
HAN	17.29%	28.75%	50.21%	68.02%
IntentGC	16.01%	25.22%	41.41%	54.26%
HetMatch $_{\backslash a}$	17.05%	28.13%	48.93%	67.01%
HetMatch $_{\backslash s}$	19.32%	30.28%	49.77%	65.63%
HetMatch $_{\backslash v}$	12.95%	22.91%	31.78%	41.50%
HetMatch	**20.50%**	**32.25%**	**53.22%**	**70.40%**
HetMatch $_{bid}$	16.35%	27.58%	49.35%	68.77%
HetMatch $_{item}$	16.22%	27.33%	48.99%	68.31%

5.5 Online Evaluation

To evaluate the effectiveness of our proposed method on the real-world products, we deployed our HetMatch both on the keyword suggestion tool and the automatic keyword hosting tool of Alibaba's e-commerce sponsored search platform. More details about these tools can be referred to in the part of Appendix. Table 6 presents the results of online AB tests on the keyword suggestion tool and the automatic keyword hosting tool in Alibaba's e-commerce sponsored search platform. For the keyword suggestion tool, our proposed method achieves an improvement of +4.19% in terms of the adopting rate and +5.35% in terms of the number of clicks over the online deployed term-match based model. For the automatic keyword hosting tool, our proposed method obtains an improvement of +10.89% in terms of the number of clicks over the online deployed GraphSage-based matching model.

Table 6: Online performance of compared methods in different scenes.

Scene	Baseline	Lift
Keyword suggestion tool	a term-match-based model	adopting rate + 4.19%, #click +5.35%
Keyword hosting tool	a two-layer Graph-Sage model	#click +10.89%

6 CONCLUSION

In this paper, we proposed HetMatch for the keyword matching problem based on a metapath-based heterogeneous GNN, which consists of a hierarchical structure to capture complex structures and rich semantics behind heterogeneous information networks in a robust way. The proposed model leverages node-level fusion, subgraph-level fusion, and Siamese neighbor matching to adaptively aggregate relevant neighborhood patterns. By introducing the autoencoder-based graph convolution layer and the Siamese matching layer, HetMatch can mitigate the negative effect from noisy patterns and enhance relevant information. In addition, we use a multi-view framework to learn the posterior probability of various objectives, which helps introduce more supervised signals. Experimental results show that our model consistently outperforms competitive approaches. Extra online AB tests also demonstrate the superiority over existing methods deployed online.

For future work, we'd like to continue exploring into other sophisticated relations hidden in the heterogeneous graph, such as text similarity, user profiles, etc. Furthermore, HetMatch still relies on human-crafted metapaths while recently there are a few transformer-based models that could potentially learn all aggregation strategies by themselves [10], which is an auspicious direction. Last, considering Bert-like pre-trained models have pushed state of the art in many areas of NLP, combining Bert with GNN to enhance the quality of keyword retrieval may also be promising. That being said, both transformer-based HGNN and Bert-like models are usually of very large model size, which would be a great challenge for training and deployment in industrial systems. Further efforts like model compression might also be required.

APPENDIX

Hyperparameter Setting

For the implementations of all comparable methods, we set the same hyperparameters.

Table 7: Hyperparameter settings.

Hyperparameter	value	Hyperparameter	value
learning rate	0.03	**optimizer**	Adam
mini-batch size	512	**#epoch**	5
latent feature size l	16	**hidden size** d	64
#neighbor m	10	**#Siamese neighbor** κ	3
#negative samples	5		

Details of Advertiser Tools for Online AB tests

Keyword Suggestion Tool. This is a suggestion tool for advertisers when they manually add keywords. More than 30% of new keywords with impressions are brought by the suggestion tools each day. For each ad, the suggestion tool will present hundreds of keyword candidates; the advertisers can adopt the candidates from the suggestion list.

The control group contains a relevance-based keyword matching model. In the treatment group, we combine keywords retrieved by HetMatch and the relevance-based model. The adopting rate of the suggested keywords and the total clicks brought by the suggested keywords are the main metrics that we are concerned about.

Keyword Hosting Tool. In Alibaba's e-commerce sponsored search platform, advertisers are provided with many black-box marketing tools such as the automatic keyword hosting tool. By merely setting budgets and bid prices, advertisers can handle online marketing for sponsored search without explicitly choosing any keywords. The automatic hosting procedure hidden behind the product can be divided into two parts. The first part is the addition of keywords based on the retrieved keywords with estimated ad clicks, while another part is to delete existing keywords that cannot bring impressions or have very low ctr/cvr.

In the experiment, the control group contains a two-layer Graph-Sage based matching model, which is proven effective in the previous AB test. In the treatment group, we use the keywords retrieved by HetMatch. We use the averaged ad clicks brought by daily keyword addition to evaluate different methods' performance.

REFERENCES

[1] Vibhanshu Abhishek and Kartik Hosanagar. 2007. Keyword generation for search engine advertising using semantic similarity between terms. In *ICEC*. 89–94.

[2] Kevin Bartz, Vijay Murthi, and Shaji Sebastian. 2006. Logistic regression and collaborative filtering for sponsored search term recommendation. In *Second workshop on sponsored search auctions*, Vol. 5.

[3] Yifan Chen, Gui-Rong Xue, and Yong Yu. 2008. Advertising Keyword Suggestion Based on Concept Hierarchy. In *WSDM*. 251–260.

[4] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*. 135–144.

[5] Wei Feng and Jianyong Wang. 2012. Incorporating heterogeneous information for personalized tag recommendation in social tagging systems. In *SIGKDD*. 1276–1284.

[6] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*. 1797–1806.

[7] Ariel Fuxman, Panayiotis Tsaparas, Kannan Achan, and Rakesh Agrawal. 2008. Using the wisdom of the crowds for keyword generation. In *WWW*. 61–70.

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

[9] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*. 1531–1540.

[10] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW*. 2704–2710.

[11] 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 *SIGKDD*. 2553–2561.

[12] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. 2333–2338.

[13] Amruta Joshi and Rajeev Motwani. 2006. Keyword generation for search engine advertising. In *ICDM-Workshops*. 490–496.

[14] Yiding Liu, Yulong Gu, Zhuoye Ding, Junchao Gao, Ziyi Guo, Yongjun Bao, and Weipeng Yan. 2020. Decoupled Graph Convolution Network for Inferring Substitutable and Complementary Items. In *CIKM*. 2621–2628.

[15] Abhirup Nath, Shibnath Mukherjee, Prateek Jain, Navin Goyal, and Srivatsan Laxman. 2013. Ad impression forecasting for sponsored search. In *WWW*. 943–952.

[16] Xichuan Niu, Bofang Li, Chenliang Li, Rong Xiao, Haochuan Sun, Hongbo Deng, and Zhenzhong Chen. 2020. A Dual Heterogeneous Graph Attention Network to Improve Long-Tail Performance for Shop Search in E-Commerce. In *SIGKDD*. ACM, 3405–3415.

[17] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *WSDM*. 459–467.

[18] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*. 593–607.

[19] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2016. A survey of heterogeneous information network analysis. *TKDE* 29, 1 (2016), 17–37.

[20] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD Explorations* 14, 2 (2012), 20–28.

[21] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Path-Sim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *VLDB* 4, 11 (2011), 992–1003.

[22] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[23] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*, Vol. 307. 1096–1103.

[24] Xuerui Wang, Andrei Broder, Marcus Fontoura, and Vanja Josifovski. 2009. A search-based method for forecasting ad impression in contextual advertising. In *WWW*. 491–500.

[25] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.

[26] Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. 2020. Graph Neural Networks in Recommender Systems: A Survey. *CoRR* abs/2011.02260 (2020). arXiv:2011.02260 https://arxiv.org/abs/2011.02260

[27] Jiarong Xu, Yang Yang, Chunping Wang, Zongtao Liu, Jing Zhang, Lei Chen, and Jiangang Lu. 2020. Robust Network Enhancement from Flawed Networks. *TKDE* (2020).

[28] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *RecSys*. 347–350.

[29] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *WWW*. 793–803.

[30] Ying Zhang, Weinan Zhang, Bin Gao, Xiaojie Yuan, and Tie-Yan Liu. 2014. Bid keyword suggestion in sponsored search based on competitiveness and relevance. *Information processing & management* 50, 4 (2014), 508–523.

[31] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. IntentGC: A Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. In *SIGKDD*. 2347–2357.

[32] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Financial Defaulter Detection on Online Credit Payment via Multi-view Attributed Heterogeneous Information Network. In *WWW*. 785–795.

[33] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. In *SIGKDD*. 1399–1407.