A Dual Heterogeneous Graph Attention Network to Improve Long-Tail Performance for Shop Search in E-Commerce

Xichuan Niu[1], Bofang Li[2], Chenliang Li[3†], Rong Xiao[2], Haochuan Sun[2], Hongbo Deng[2], Zhenzhong Chen[1]
[1]School of Remote Sensing and Engineering, Wuhan University, China
[2]Alibaba Group, Hangzhou, China
[3]Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China
[1]{niuxichuan, zzchen}@whu.edu.cn, [2]{bofang.lbf, xiaorong.xr, haochuan.shc, dhb167148}@alibaba-inc.com
[3]cllee@whu.edu.cn

ABSTRACT

Shop search has become an increasingly important service provided by Taobao, the China's largest e-commerce platform. By using shop search, a user can easily identify the desired shop that provides a full-scale of relevant items matching his information need. With the tremendous growth of users and shops, shop search faces several unique challenging problems: 1) many shop names do not fully express what they sell, i.e., the semantic gap between user query and shop name; 2) due to the lack of user interactions, it is difficult to deliver a good search result for the long-tail queries and retrieve long-tail shops that are highly relevant to a query.

To address these two key challenges, we resort to graph neural networks (GNNs) which have various successful applications in arbitrarily structured graph data. Specifically, we propose a dual heterogeneous graph attention network (DHGAT) integrated with the two-tower architecture, using the user interaction data from both shop search and product search. At first, we build a heterogeneous graph in the context of shop search, by exploiting both the first-order and second-order proximity from *user search behaviors, user click-through behaviors* and *user purchase records*. Then, DHGAT is devised to attentively adopt heterogeneous and homogeneous neighbors of query and shop to enhance representations of themselves, which can help relieve the long-tail phenomenon. Besides, DHGAT enriches semantics of query text and shop name by compositing the titles of the relevant items to alleviate the semantic gap. Moreover, to enhance the graph representation learning, we augment DHGAT with a regularized neighbor proximity loss (NPL) to explicitly learn the graph topological structure and train whole framework in an end-to-end fashion. Compelling results from both offline evaluation and online A/B tests demonstrate the superiority of DHGAT over state-of-the-art methods, especially for long-tail queries and shops.

CCS CONCEPTS

• **Information systems** → **Document filtering**; **Retrieval models and ranking**.

KEYWORDS

Shop Search, Graph Neural Networks, E-commerce

ACM Reference Format:
Xichuan Niu, Bofang Li, Chenliang Li, Rong Xiao, Haochuan Sun, Hongbo Deng, Zhenzhong Chen. 2020. A Dual Heterogeneous Graph Attention Network to Improve Long-Tail Performance for Shop Search in E-Commerce. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403393

1 INTRODUCTION

Search engine is the key functionality in driving rapid growth of e-commerce platforms, such as Taobao[1] and Amazon[2]. Due to the large volume of users and items, it is challenging to deliver satisfied results that meet information needs of customers. Therefore, great efforts have been made to enhance the performance of e-commerce search engine, which plays a significant role in improving user experience and promoting the sales revenue of online merchants. Specifically, a shop in an e-commerce platform often sells items relevant to a specific category, e.g., a shop sells all kinds of sportswear. Also, the promotion campaign is often launched in a shop-wide manner, where a larger discount could be offered with a bigger order. Hence, it becomes more convenient for a user to purchase relevant items in a single shop and enjoy an overlaid promotion. Nowadays, shop search has become an important service for Taobao platform. To our knowledge, no previous effort has been made towards improving the performance of shop search in e-commerce.

Shop search, however, confronts several distinct challenges. Firstly, the user queries and shop names are often very short in nature. There exists semantic gap between user queries and shop names.

† Chenliang Li is the corresponding author.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403393

[1]http://taobao.com/
[2]https://www.amazon.com/

That is, regularly user queries are more casual while sellers describe their shops more formally. And sometimes a shop name can not reflect what they sell or serve completely, which causes more difficulties to conform user's intent. So simple lexical matching very likely falls short in these cases. Besides, unlike product search, user's shop search behavior and the feedback signals (clicks or purchases) are quite sparser. Note that a user can directly search for interested items rather than shops. Also, they can access the desired shops through previous orders or favorites collection. It is widely known that most of the exposures are popular shops or queries of high frequency. For the long-tail queries and shops, we usually do not have adequate feedback data to train a ranking model, leading to the long-tail bias. These hot shops could domainte the search results and hence degrade the user experience. Moroever, when a long-tail query is issued, it could be difficult to generate a good ranking result.

It has been shown that modelling the e-commerce interactions as structured graph attains significant results. [25] generates node sequence through random walk and the Skip-Gram model [20] is used to train graph embeddings. Then the learned vertice representations are further injected into click through rate (CTR) prediction task. [29] leverages DeepWalk [21] and Node2vec [11] embedding techniques to integrate click-graph features into a unified neural ranking framework also in a two-stage manner. These two-stage approaches are not directly designed for search optimization targets (e.g. CTR, CVR), since the graph based representation learning works as a separate component. This pipelined optimization scheme leads to inferior expressive ability for e-commerce search scenes.

Recently, Graph Neural Networks (GNNs) have achieved magnificent success in various natural language processing and information retrieval tasks [26, 29, 30]. The aim of GNNs is to generalize neural network algorithms to non-Euclidean domains (such as graphs) for robust feature learning. GNNs mainly derive the node representation by aggregating the features appearing in the neighborhood. That is, the underlying graph structure determines how much information can be assimilated in order to facilitate the representation of node itself. Several works have utilized GNNs to deliver promising performance for ranking and recommendation applications [3, 25, 28]. It is expected that modeling homogeneous and heterogeneous relations on the basis of various user interactions and rich semantic information in e-commerce would provide a promising avenue to enhance long-tail performance of shop search.

To this end, in this paper, we propose a heterogeneous graph neural network for shop search in e-commerce. Specifically, we develop a dual heterogeneous graph attention network to boost the representation learning of queries and shops, named DHGAT. The main goal of DHGAT is to enrich the represenations of queries and shops by exploiting various kinds of semantic signals in the e-commerce platform. Here, we build a heterogeneous graph in the context of shop search by exploiting the first-order and second-order proximity from three kinds of user interactions: *user search behaviors, user click-through bahaviors* and *user purchase records*. Both the homogeneous and heterogeneous relations included in the graph can help us derive robust graph-based representations for queries and shops. By introducing a hierarchical attention mechanism, DHGAT can selectively identify the relevant homogeneous and heterogeneous neighbors to update the representation for self node. To further bridging the semantic gap incurred by the mismatch between queries and shops, DHGAT exploits the semantic textual information provided by the relevant items to derive text-level representations for queries and shops. At last, both graph-based representations and text-level representations along with multiple user features are composited and fed into a two-tower network for relevance estimation. For model training, we introduce a regularized neighbor proximity loss as an auxiliary task to enhance the representation learning to retain the graph topological structure, leading to better search performance.

The main contributions of our work can be summarized as follows:

(1) We propose a dual hierarchical graph attention network for e-commerce shop search task. A heterogeneous graph is constructed to perform graph based representation learning for both shops and queries, which includes both first-order and second-order proximity from various user interactions in e-commerce. To the best of our knowledge, this is the first attempt to address the task of shop search in e-commerce.

(2) The proposed DHGAT selectively leverages heterogeneous and homogeneous relations of queries and shops simultaneously to derive the robust semantic representations for shop search, especially for long-tail performance enhancement. The textual information provided by the relevant items is also exploited to further alleviate the semantic gap.

(3) To explicitly learn the topological structure within graph, we add a regularized neighbor proximity loss to main CTR prediction loss. The whole framework is trained in an end-to-end fashion. Offline evaluation and online A/B test results prove the effectiveness and efficiency of our proposed model.

2 PRELIMINARIES

In this section, we first introduce the problem setup and describe heterogeneous graph. Then we give a detailed description towards the graph construction in context of shop search in e-commerce.

2.1 Problem Setup

Shop Search In E-commerce. Given a set $< \mathcal{U}, \mathcal{Q}, \mathcal{S} >$, where $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ denotes the set of m users, $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$ denotes the set of n queries, $\mathcal{S} = \{s_1, s_2, ..., s_p\}$ denotes the set of p shops. When a user u issues a query q, the shop search engine is required to retrieve the most relevant shops as a ranking list where the more relevant shops are ranked higher (i.e., with a higher relevance score).

In context of shop search in e-commerce, there are various kinds of entities (i.e., users, items, shops and queries) and interactions between them. It is natural to model this complex system as a heterogeneous graph, where user, query, shop, item are taken as different types of nodes, and the edges between them reflect the relatedness in terms of the associated interactions.

Heterogeneous Graph. A heterogeneous graph is defined as a network $G = (\mathcal{V}, \mathcal{E})$ consisting of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$. It is also associated with a node type mapping function $\phi(v)$:

$\mathcal{V} \rightarrow \mathcal{O}$ and an edge type mapping function $\psi(e) : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{O}$ and $\mathcal{R}$ denote the set of all node types and the set of all edge types respectively and $|\mathcal{O}| + |\mathcal{R}| > 2$. Here, an edge type indicates the connection between two particular types of nodes.

As mentioned above, for shop search, we can build a heterogeneous graph to include all related types of entities as nodes. For instance, a query node can have homogeneous query neighbors, heterogeneous shop and item neighbors. We plan to take advantage of the information encoded in the heterogeneous graph to facilitate the task of shop search and improve user experience. Given a heterogeneous graph $G = (\mathcal{V}, \mathcal{E})$, we devise a novel heterogeneous graph representation learning framework to embed each node as a dense vector representation: $\mathcal{F}_\Theta : \mathcal{V} \rightarrow \mathbb{R}^{|\mathcal{V}| \times d}$, where d is the dimension size and $d \ll |\mathcal{V}|$, and Θ denotes the learnable parameters. Then, for a given shop and query pair, the corresponding representations are fed into a neural network for relevance estimation.

2.2 Graph Construction

Since the efficacy of the graph representation learning highly relies on the underlying graph structure, a key step is to construct a heterogeneous graph that precisely holds the semantic relatedness between different entities in context of shop search. Here, we first collect different kinds of user interactions in the e-commerce platform as the raw data for graph construction. These interactions include *user search behaviors, user click-through behaviors, user purchase records*. Specifically, there are three types of nodes in the constructed graph: Shop (S), Query (Q) and Item (I). By exploiting the first-order and second-order proximity from the raw interaction data, we build three edge types for these three node types correspondingly, which cover both *homogeneous* and *heterogeneous* neighbors for each node.

Homogeneous Neighbors. The homogeneous neighbors refer to the nodes of the same type. This kind of relations often reveal the shared characteristics between the two nodes to some extent. For example, two queries that share the same search intent, or two shops that provide very similar items. For queries and shops, we utilize the following rules of second-order proximity to establish the homogeneous neighbors: 1) users mostly follow a changeless purpose within a search session. Hence, the queries belonging to same session constitute each other's homogeneous neighbors; 2) the queries that lead to the click of same shop often reveal overlap of user's search intent. Hence, the queries matching this criterion are formulated as each other's homogeneous neighbors; 3) the shops that are clicked under the same query are likely to function similarly (e.g., provide the similar items). Hence, these shops are considered as each other's homogeneous neighbors.

Heterogeneous Neighbors. For heterogeneous neighbors, first of all, we make use of click-through data for shop search directly. That is, the shops clicked under a query become the query's heterogeneous shop neighbors and vice versa. Note that this first-order proximity offered by the shop search click-through data could be very sparse since a large proportion of shops are not displayed. The resultant heterogeneous relations could not fully cover the semantic relatedness between shops and queries, leading to inferior long-tail performance.

Therefore, to enrich the heterogeneous query-shop relations extracted through the shop search click-through data, we propose to transfer knowledge from product search, the main search engine for most e-commerce platforms. Due to the large amount of product search click-through data, we can leverage the second-order query-shop relations to augment the corresponding first-order relations. Specifically, for an item provided by a shop and purchased under a query, the corresponding shop and query are considered as each other's heterogeneous neighbor respectively.

Note that exploiting the second-order query-shop relations from the product search click-through data may not alleviate the data sparsity problem to its fullness, because the long-tail phenomenon still exists in the product search scenario. Moreover, the name of a shop is very short due to the restriction made by the platform. It is hard to express all the items sold in the shop by a short shop name. It is expected to alleviate long-tail problem by encoding the semantic information of the constituent items to bridge this semantic gap. Hence, we propose to further exploit the textual semantics covered by the item titles. For each query, we treat the items sold under it as the former's heterogeneous item neighbors. Similarly, the items belonging to the same shop are considered as the shop's heterogeneous item neighbors. As will be discussed in the next section, the proposed DHGAT exploits the item neighbors and their associated titles to derive the text-level representations for queries and shops.

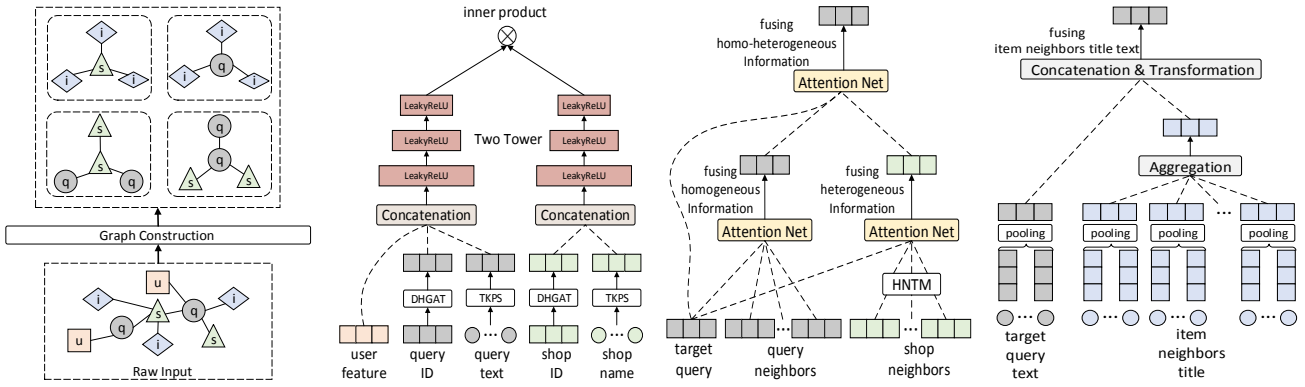
After establishing the edges between nodes, we further assign weight to each edge to indicate the relatedness between the two nodes. For the edge between each shop and item pair, the edge weight is set to be 1. For the other edge types, the edge weight is simply set to be the number of corresponding user interactions. An illustration for graph construction is shown in Figure 1(a), where the homogeneous and heterogeneous neighbors derived from the user interactions are shown as the bottom two after graph construction, while the product search based item neighbors are shown as the upper two.

3 METHOD

In this section, we formally present our proposed model DHGAT to tackle the challenges mentioned in Section 1. For each part, we start from the motivations and intuitions, and then introduce the technical details. Next, we present objective function and some training techniques that can enhance the model performance.

3.1 Overview

The framework architecture of DHGAT is illustrated in Figure 1(c). The basic idea of the proposed model DHGAT is to design a heterogeneous GNN for enriching the representations of queries and shops. As shown in Figure 1(b), our model takes a triple set $< u, q, s >$ as input, where u and q stands for user u issues a query q, s represents a candidate shop. We samples by weight a fix-sized set of neighbors for query and shop respectively, including homogeneous ones and heterogeneous ones. Different types of neighbors are aggregated through a hierarchical attention layer to strengthen ID embedding representations. Dual heterogeneous graph attention networks are operated for queries and shops at the same time. Moreover, borrowing item transaction history from product search



(a) Graph Construction (b) Two Tower (c) DHGAT (d) TKPS

Figure 1: The whole pipeline of our model: (a) Graph Construction, (b) Two Tower Architecture, (c) Dual Heterogeneous Graph Attention Network, (d) Transferring Knowledge from Product Search.

(Figure 1(d)), we clarify and enhance semantics of query text and shop name by exploiting their heterogeneous item neighbors' title. Multiple user features are incorporated to yield personalized search results. At last, user-query representations and shop representations are sent into a tow-tower architecture to get predicted score, according to which a sorted shop list will be displayed.

3.2 Dual Heterogeneous Graph Attention Network

The key challenge of applying GNNs to shop search is how to compose homogeneous and heterogeneous neighbors of queries and shops. Firstly we denote query's homogeneous neighbors as $\mathcal{N}_o(q)$, query's heterogeneous neighbors as $\mathcal{N}_e(q)$, similarly $\mathcal{N}_o(s)$, $\mathcal{N}_e(s)$ are represented as shop's homogeneous neighbors and shop's heterogeneous neighbors respectively. Since some hot queries and shops may have tons of graph neighbors, we begin by sampling $2N$ neighbors for each query and shop, N homogeneous ones and N heterogeneous ones. Sampling weights are calculated and normalized as stated in Section 2.2.

After getting sampled neighbors for queries and shops, a natural idea is to aggregate neighborhood information for reinforcing self node representation. Take query node as an example:

$$\mathbf{h}^t_{\mathcal{N}(q)} = AGGREGATE(\{\mathbf{h}^{t-1}_v, \forall v \in \mathcal{N}_o(q) \cup \mathcal{N}_e(q)\}) \quad (1)$$

$$\mathbf{h}^t_q = COMBINE(\mathbf{h}^{t-1}_q, \mathbf{h}^t_{\mathcal{N}(q)}) \quad (2)$$

where $\mathbf{h}^t_q$ is ID embedding of query node q at layer t, $AGGREGATE$ denotes neighbor aggregation function, such as averaging or max-pooling operation, and $COMBINE$ denotes the function that merges the aggregated neighborhood representation with the node's representation.

However, we have both homogeneous query and heterogeneous shop neighbors for query node, simply adopting one overall aggregation may lose much information. We devise a hierarchical attentive aggregation strategy to discriminate the influence of different types of neighbors. Specifically, we resort to a two-level attention mechanism, where the first level focuses on aggregating neighbor nodes within each type and the second level targets at aggregation over different types. For query node q, the first level attention over homogeneous query neighbors is formulated as:

$$\mathbf{h}^t_{\mathcal{N}_o(q)} = \sum_{i \in \mathcal{N}_o(q)} \alpha^t_{q,i} \mathbf{h}^{t-1}_i \quad (3)$$

where $\alpha^t_{q,i}$ is the normalized attention weight of homogeneous neighbor node i at layer t:

$$\alpha^t_{q,i} = \frac{exp(LeakyReLU(\mathbf{a}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathbf{h}^{t-1}_i]))}{\sum_{k \in \mathcal{N}_o(q)} exp(LeakyReLU(\mathbf{a}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathbf{h}^{t-1}_k]))} \quad (4)$$

where $\mathbf{a}_{q,t}$ stands for each layer using its own attention parameters, the operator $\|$ denotes concatenation, and we use $LeakyReLU$ as the activation function.

Likewise, for query node q, its heterogeneous shop neighbors are combined at the first level attention. But as query embedding and shop embedding don't share the same vector space, we propose to add a heterogeneous neighbor transformation matrix (HNTM) before the attention layer:

$$\mathbf{h}^t_{\mathcal{N}_e(q)} = \sum_{j \in \mathcal{N}_e(q)} \alpha^t_{q,j} \mathcal{W}_{QS} \mathbf{h}^{t-1}_j \quad (5)$$

where $\mathcal{W}_{QS}$ is the semantic transformation matrix for query's heterogeneous shop neighbors, and $\alpha^t_{q,j}$ is calculated as:

$$\alpha^t_{q,j} = \frac{exp(LeakyReLU(\mathbf{b}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathcal{W}_{QS}\mathbf{h}^{t-1}_j]))}{\sum_{k \in \mathcal{N}_e(q)} exp(LeakyReLU(\mathbf{b}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathcal{W}_{QS}\mathbf{h}^{t-1}_k]))} \quad (6)$$

where $\mathbf{b}_{q,t}$ is the attention parameter at layer t for measuring weights of heterogeneous neighbors.

Furthermore, after obtaining $\mathbf{h}^t_{\mathcal{N}_o(q)}$ and $\mathbf{h}^t_{\mathcal{N}_e(q)}$, we apply the second attention to fuse both homogeneous and heterogeneous information as followed:

$$\mathbf{h}^t_{\mathcal{N}(q)} = \beta^t_{\mathcal{N}_o(q)} \mathbf{h}^t_{\mathcal{N}_o(q)} + \beta^t_{\mathcal{N}_e(q)} \mathbf{h}^t_{\mathcal{N}_e(q)} \quad (7)$$

$$\beta^t_{\mathcal{N}_*(q)} = \frac{exp(LeakyReLU(\mathbf{c}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathbf{h}^t_{\mathcal{N}_*(q)}]))}{\sum_{k=\{\mathcal{N}_o(q),\mathcal{N}_e(q)\}} exp(LeakyReLU(\mathbf{c}^\top_{q,t}[\mathbf{h}^{t-1}_q \| \mathbf{h}^t_k]))} \tag{8}$$

where $\mathbf{c}_{q,t}$ is the parameter of second level attention at layer t, $\beta^t_{\mathcal{N}_*(q)}$ is the attention weights to differentiate two types (i.e., homogeneous and heterogeneous) of neighbors.

$\mathbf{h}^t_{\mathcal{N}(q)}$ is the aggregated neighborhood representation as a result of hierarchical attention. Then we do the *COMBINE* operation regarding to $\mathbf{h}^t_{\mathcal{N}(q)}$ and $\mathbf{h}^{t-1}_q$. First we concatenate them and employ a linear transformation:

$$\mathbf{h}^t_q = LeakyReLU(\mathcal{W}^t_q \cdot [\mathbf{h}^{t-1}_q \| \mathbf{h}^t_{\mathcal{N}(q)}]) \tag{9}$$

where $\mathcal{W}^t_q$ is the learnable transformation parameter. $\mathbf{h}^t_q$ is the query node representation at layer t that has fused both homogeneous and heterogeneous information.

On the other hand, for shop node s, the hierarchical attention mechanism works in an identical way, except the attention parameters are distinct, and the semantic transformation matrix should change from $\mathcal{W}_{QS}$ to $\mathcal{W}_{SQ}$.

3.3 Transferring Knowledge from Product Search

The reason why we need to transfer knowledge from product search (TKPS) is three fold: (1) frequently user queries are ambiguous and shop names may don't reflect what they sell, so we don't aggregate neighbor's texts through hierarchical attention framework since much noise may be carried by them, (2) user behaviors of shop search are fairly sparser while transaction history of product search contains richer information, (3) incorporating items (sold under a query or owned by a shop) title texts can contribute to mitigating the semantic gap between user queries and shop names.

We adopt uniform term embedding scheme described in [9], which represents queries, shops and items with one set of word term embeddings. As shown in Figure 1(d), still take a piece of query q composed of n terms as example, let its terms be $(w_{q,1}, w_{q,2}, ..., w_{q,n})$. At the embedding layer, each term is converted into a low-dimensional dense vector representation and q is expressed as $(\mathbf{t}_{q,1}, \mathbf{t}_{q,2}, ..., \mathbf{t}_{q,n})$. Then we derive the text representation of query q from its term embeddings:

$$\mathbf{e}_q = f(\mathbf{t}_{q,1}, \mathbf{t}_{q,2}, ..., \mathbf{t}_{q,n}) \tag{10}$$

where f means the operation function applied to the terms. In our experiments, we adopt the average function. Similarly, text representations of shop and item are derived as $\mathbf{e}_s, \mathbf{e}_i$ respectively.

Afterwards we non-uniformly sample N item neighbors for query q according to the weights defined in Section 2.2, denoted as $\mathcal{N}_m(q)$. We aggregate the item neighbors text as followed:

$$\mathbf{e}^t_{\mathcal{N}_m(q)} = AGGREGATE(\{\mathbf{e}^{t-1}_i, \forall i \in \mathcal{N}_m(q)\}) \tag{11}$$

where $\mathbf{e}^{t-1}_i$ is item text representation vector at layer $t-1$ and $\mathbf{e}^0_i = \mathbf{e}_i$. Here instead of attentive aggregation, we design the *AGGREGATE* function as mean pooling. Owing to the fact that query itself may hold vague meaning, so query-guided attention may bring in mistaken information.

Then we combine $\mathbf{e}^{t-1}_q$ and $\mathbf{e}^t_{\mathcal{N}_m(q)}$ to get $\mathbf{e}^t_q$:

$$\mathbf{e}^t_q = LeakyReLU(\mathcal{W}^t_{qe} \cdot [\mathbf{e}^{t-1}_q \| \mathbf{e}^t_{\mathcal{N}_m(q)}]) \tag{12}$$

where $\mathcal{W}^t_{qe}$ is the parameter matrix. Correspondingly we can acquire text representation of shop s at layer t: $\mathbf{e}^t_s$, which fuses textual message of shop names and item neighbor titles.

3.4 Incorporating User Features

To better characterize users and retrieve personalized search results, we augment the DHGAT model with additional user features. In our framework, user features are represented in a multi-field multi-hot encoding form. Each field contains multiple discrete categorical features or bag-of-words (BoW) features that are analogous semantically (details of all user features can be found in Appendix A.4). For example, one input instance has three features: *[gender=male, city=beijing, interests=cook&running]*. It is translated into several high-dimensional sparse features via field-aware one-hot encoding:

$$\underbrace{[0,1]}_{gender}\ \underbrace{[1,0,0,0,...,0]}_{city}\ \underbrace{[0,1,...,0,0,1]}_{interests}$$

Then the raw sparse features are fed into the embedding layer to generate low-dimensional real-valued dense vector. The feature embedding is used as the field embedding when the field is single-valued. And if the field is multi-valued, we apply mean pooling to the feature embeddings to get the field embedding. The result of embedding layer $\mathbf{u}$ is a wide concatenated vector of all fields.

3.5 Two-tower Architecture

The two-tower model architecture is shown in Figure 1(b). As can be seen, we first concatenate DHGAT output of query $\mathbf{h}_q$ (fusing homogeneous and heterogeneous neighbor IDs) and TKPS output of query $\mathbf{e}_q$ (fusing item neighbor titles), then combine the concatenation vector and user feature embedding $\mathbf{u}$, which is fed into left tower for more feature fusions. The right tower accepts as input concatenation result of $\mathbf{h}_s$ and $\mathbf{e}_s$ with regard to shop. In our experiments, two-tower are implemented as multiple layer perceptrons (MLPs) whose hidden neurons decrease as the height rises.

3.6 Objective and Model Training

At the last layer L of two-tower, we obtain user-query representation and shop representation $\mathbf{h}^L_{uq}$ and $\mathbf{e}^L_s$. The probability of user u clicking shop s after issuing a query q can be predicted by:

$$\hat{y}_{uqs} = nn(\mathbf{h}^L_{uq}, \mathbf{e}^L_s) \tag{13}$$

For implicit feedback, $nn(\cdot)$ can be a fully-connected layer with a sigmoid activation function. In practice, taking into account efficiency and scalability, we utilize the simplest vector inner product.

The loss function measures the discrepancy between predicted value $\hat{y}_{uqs}$ and ground-truth value y_{uqs}. Here we adopt the most widely used loss function in CTR prediction task, i.e., binary cross entropy:

$$\mathcal{L}_1 = -\sum_{(u,q,s)} y_{uqs} log(\hat{y}_{uqs}) + (1 - y_{uqs}) log(1 - \hat{y}_{uqs}) \quad (14)$$

Besides, using CTR (i.e. click or not click) as the sole supervisory signal to train the model may not capture those information encoded in graph topological structure effectively. We are required to explicitly involve graph neighbor relations in the loss function. Hence, we add a novel neighbor proximity loss to the main CTR prediction loss, which draws closer homogeneous neighbors in the high dimensional space and pushes farther randomly sampled negative nodes:

$$\mathcal{L}_2(\mathbf{h}_i) = -log(\sigma(\mathbf{h}_i^\top \mathbf{h}_j)) - \sum_{k \sim P(i)}^{N_{neg}} log(\sigma(-\mathbf{h}_i^\top \mathbf{h}_k)) \quad (15)$$

where $\mathbf{h}_i, \mathbf{h}_j$ and $\mathbf{h}_k$ denotes embeddings of the current node (query or shop), the homogeneous neighbor node in a pair, and the randomly sampled negative node. N_{neg} is the number of sampled negative nodes, we leave the concrete details of negative sampling in Appendix A.1.

Combining CTR prediction cross entropy and neighbor proximity loss, we reach the following complete loss function for DHGAT:

$$\mathcal{L} = \mathcal{L}_1 + \alpha \mathcal{L}_2 + \lambda \|\Theta\|_2^2 \quad (16)$$

where $\|\Theta\|_2^2$ is the L2-regularizer on parameters and embeddings, α and λ are balancing hyper-parameters of neighbor proximity loss and L2 regularization respectively. We use AdaGrad optimizer [8] to minimize the loss.

4 OFFLINE EXPERIMENTS

In this section, we conduct experiments on Taobao real-world dataset to evaluate our proposed model against up-to-date state-of-the-art methods. We then examine hyper-parameter sensitivity and perform the ablation study to analyze effects of different neighbor types.

4.1 Dataset

We evaluate our proposed DHGAT model based on a large-scale real world dataset collected from Taobao shop search platform, which serves over 120 million monthly active users. The dataset covers ten consecutive days of user's click records from the shop search log of Taobao mobile App during November 2019. We hold the first nine days as training set and leave the tenth day for testing, which is refered as *Normal* test set. For the purpose of transferring knowledge of product search, we collect item transaction history of the same period also from Taobao mobile App. To verify the generalization ability of the model, we filter out a subset that only consists of the pairs of query and shop that don't appear in the training set, which we call *Hard* test set. And we further construct a *Long Tail* test set to validate the strength of graph-based model on the long-tail problem, where we select those queries or shops that emerge only once in the training set. The basic statistics of training set, three test sets as well as graph neighbor relations are listed in Appendix A.3.

4.2 Experimental Settings

4.2.1 Baseline Methods. We compare our model with three categories of baseline methods: (1) classical information retrieval and semantic matching methods (NCF [13], DSSM [15], DNN), (2) two-stage graph embedding injected searching methods (GEPS [29], TGE-PS [4]), (3) end to end graph neural networks based methods (PinSage [28], GAT [24], HAN [26]). We also report the performance of two variants of our model (DHGAT, DHGAT$_{NP}$) to test the effectiveness of the components. Hyperparameter settings and implementation details for baselines and our method are introduced in Appendix A.1 and A.2.

- NCF [13]: It is the classical neural network method for top-N recommendation. Here we feed it with the query ID representation and shop ID representation instead of user and item.
- DSSM [15]: This model is proposed for web search semantic matching. We employ the latest BERT [5] to extract text features and measure the similarity of query and shop.
- DNN: Content-enhanced deep learning searching model, which concatenates ID representations with text feature embeddings and then feeds through into several MLP layers.
- GEPS [29]: A two-stage method that utilizes DeepWalk [21] and Node2vec [11] graph embedding techniques to enable neural retrieval models to exploit graph structured data for automatic feature extraction.
- TGE-PS [4]: A latest inductive graph embedding approach which uses text-driven graph embedding to generate node embeddings from rich texts.
- PinSage [28]: The state-of-the-art web-scale GNN recommender system with GraphSage [12] as the backbone GNN model. We adopt the optimal implementation released in [12] and report the results based on three neighbor aggregation techniques: Mean (PinSage_M), Max-pooling (PinSage_P), LSTM (PinSage_L).
- GAT [24]: The state-of-the-art attention-based GNN method that leverages attention mechanism to aggregate neighborhood nodes. We report the results of single head and multi-head attention (i.e., K = 8), respectively.
- HAN [26]: The state-of-the-art heterogeneous network embedding model, which takes the importance of nodes and meta-paths into consideration simultaneously.
- DHGAT$_{NP}$: It is a variant of DHGAT, which is non personalized, i.e., removes the additional user features.
- DHGAT: It is our complete model.

4.2.2 Evaluation Protocol. We adopt four commonly used performance metrics for offline evaluation: *Area Under the receiver operating characteristic Curve* (AUC), *Group AUC* (GAUC), *Mean Reciprocal Rank* (MRR), *Hit Ratio at Rank K* (HR@K). Note that GAUC is different from AUC because it can measure the discrepancy between predictions and ground truths under one specific query and combine all queries using frequency as weight. Intuitively, MRR cares about the ranking positions of positive shops while HR@K accounts for whether test shops are present in top-k list. In our experiments, we calculate the HR@1 and HR@5 to evaluate accuracy in the first few retrieval results.

4.3 Overall Performance

We evaluate baselines and our algorithm using three test datasets to explore the model's performance under different scenarios. The overall results of different methods under five indicators are presented in Table 1, from which the following observations can be made:

(1) In all three test datasets, the proposed DHGAT model significantly outperforms all the compared baselines on all five evaluation metrics. Specifically, it offers the average relative performance gain of 4.94%, 4.04%, 2.68%, 4.15%, 1.01% in AUC, GAUC, MRR, HR@1, HR@5 over the best baselines respectively. In particular, we find that the improvements of DHGAT on *Hard* and *Long Tail* are higher than *Normal* compared with baselines. The results indicate that DHGAT is able to successfully aggregate information of homogeneous and heterogeneous neighborhoods and combine item neighbors title from product search. The incorporation of graph neighbor relations helps to strengthen the representation learning of queries and shops and further enhance the prediction performance, especially when the model needs to deal with generalization and long-tail problem.

(2) Pertaining to baseline methods, we observe that DNN and NCF are actually better than DSSM, which demonstrates that classical semantic matching algorithm cannot handle the semantic gap between user queries and shop names. The two-stage graph embedding injected models (GEPS, TGE-PS) performs worse than end to end GNNs based ones. This is probably because these two methods are not directly optimized for specific CTR prediction task, thus limiting the expression ability of generated node embeddings. As for the GNNs based approaches, we find in most cases performance order on different metrics is: HAN > PinSage > GAT. It is not surprising since HAN is capable of exploiting heterogeneous relationships while the other two approaches only model homogeneous information. Note that Mean and LSTM aggregators achieve considerable gain over Max-pooling aggregator on *Hard* and *Long Tail* dataset. And mutli-head GAT(K=8) does perform better than single head version.

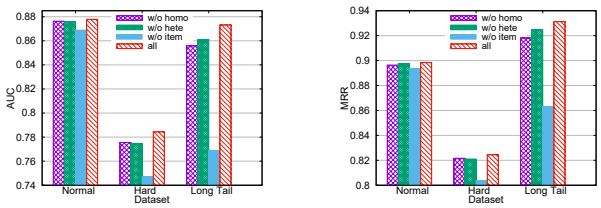
(3) We also show the results of a non personalized variant of DHGAT, i.e, DHGAT$_{NP}$. From Table 1, we can tell that DHGAT$_{NP}$ already exceeds all the baseline methods, which is attributed to its representation power of queries and shops in both identification and text. Moreover, we further augment DHGAT$_{NP}$ with additional user features to fetch personalized search results. The performance gains of DHGAT indicate that incorporation of user features is necessary for better characterizing the user and understanding his intention.

(4) As mentioned above, different evaluation metrics can crystallize different aspects of a model. DHGAT shows strong performance on all five metrics, which reveals that it not only can decide whether a shop meets user requirements, but also is able to arrange a satisfactory ordered list for display.

4.4 In-depth Analysis

To better understand the performance of the proposed model, we conduct a series of in-depth analysis on DHGAT$_{NP}$.

Differentiating the importance of neighbor types. To examine which neighbor types play important roles in predicting user's click decisions, we conduct ablation tests by removing different neighbor types. *w/o homo, w/o hete, w/o item* and *all* denote removing query-query edges & shop-shop edges, removing query-shop edges & shop-query edges, removing query-item edges & shop-item edges and the complete model, respectively. We only report the results with respect to AUC and MRR as similar performance patterns are also observed for other metrics.



(a) AUC (b) MRR

Figure 2: Effects of different neighbor types.

As is shown in Figure 2, removing any kind of neighbor will lead to poor performance on both metrics, which means that all three neighbor types contribute to enhancing representation learning. On the other hand, DHGAT assigns highest importance to item neighbor, proving the necessity of transferring knowledge from product search. Homogeneous and heterogeneous neighbors are nearly equally important on *Normal* and *Hard* dataset, nevertheless, homogeneous neighbors appears more critical on *Long Tail* dataset. One plausible explanation is that when query or shop is very unpopular, its possible popular homogeneous neighbors determine more.

Hyperparameter sensitivity. Figure 3 and Figure 4 are illustration of hyperparameter sensitivity of neighbor count and α with respect to AUC and MRR on all three test datasets. Note that the x-axis of Figure 4 is logarithmic.

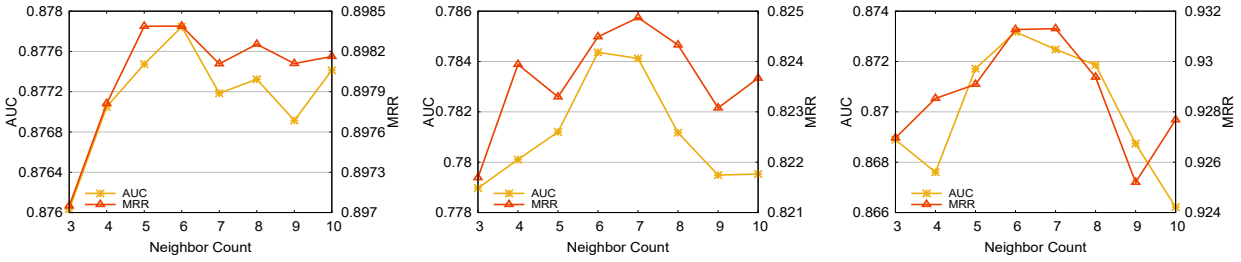
As is depicted in the two figures, we vary the sampling neighbor count from 3 to 10 and α from 0 to 1 while keeping other parameters fixed. The yellow line represents the AUC value, and the red line indicates the MRR metric. From Figure 3, we observe that the best performance is obtained when neighbor count is 6 or 7. Either sampling more or less neighbors will result in performance degradation to some extent. This is reasonable because too few neighbors carry insufficient information while too many would inevitably introduce much noise. Similarly, we see from Figure 4 that free of neighbor proximity loss (i.e., $\alpha = 0$) causes the model to perform inferiorly while pushing too much weight on it (i.e., $\alpha = 1$) also gives rise to worse results. Hence, we set $\alpha = 0.001$ in our experiments.

5 ONLINE EXPERIMENTS

We deploy our proposed model DHGAT on the search matching stage of shop search in Taobao. The graph neighbor relations cover search logs of all training days, and can be updated incrementally each day. Under the framework of A/B tests (i.e., bucket tests), one bucket is selected for baseline and another bucket for our model. For whole bucket, the goal is to increase user activity and maximize the revenue. DHGAT achieves performance gains of **2.506%** on

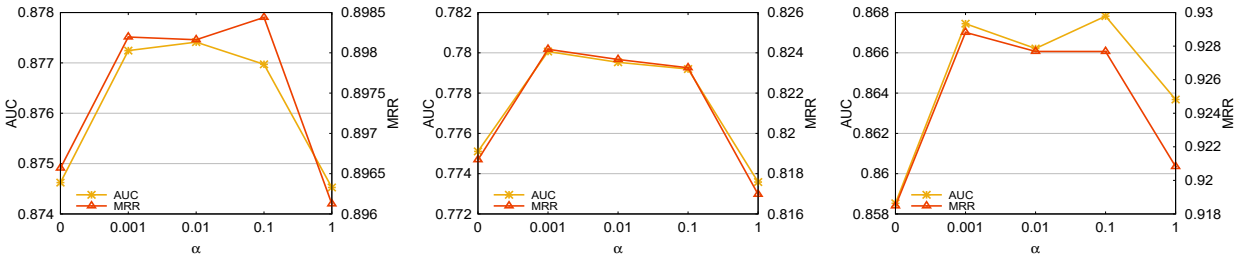
Table 1: Performance comparison for baselines and DHGAT. The * indicates the best performance of the baselines. Best results of all methods are highlighted in boldface. Improvement over the best baseline are shown in the last row.

Method	Normal					Hard					Long Tail				
	AUC	GAUC	MRR	HR@1	HR@5	AUC	GAUC	MRR	HR@1	HR@5	AUC	GAUC	MRR	HR@1	HR@5
NCF	0.8657	0.8517	0.8930	0.7621	0.9289	0.7326	0.7277	0.8040	0.5896*	0.8831	0.7237	0.6581	0.8563	0.8841	0.9926
DSSM	0.8076	0.7871	0.8363	0.6320	0.8938	0.6909	0.6810	0.7524	0.4774	0.8503	0.6968	0.6424	0.8277	0.8596	0.9927
DNN	0.8651	0.8522	0.8920	0.7590	0.9300*	0.7519*	0.7370*	0.8037	0.5847	0.8880*	0.8090*	0.7086*	0.8876*	0.9073*	0.9943*
GEPS	0.8656	0.8521	0.8932	0.7619	0.9274	0.7304	0.7265	0.8036	0.5886	0.8826	0.7187	0.6539	0.8477	0.8777	0.9924
TGE-PS	0.8645	0.8513	0.8918	0.7610	0.9268	0.7282	0.7199	0.8003	0.5855	0.8830	0.7174	0.6542	0.8467	0.8758	0.9921
PinSage_M	0.8670	0.8521	0.8928	0.7618	0.9275	0.7406	0.7267	0.8007	0.5836	0.8768	0.7758	0.6770	0.8690	0.8938	0.9935
PinSage_P	0.8647	0.8510	0.8926	0.7615	0.9265	0.7279	0.7244	0.8029	0.5864	0.8818	0.7348	0.6655	0.8660	0.8897	0.9936
PinSage_L	0.8665	0.8525	0.8929	0.7614	0.9283	0.7393	0.7270	0.8009	0.5840	0.8794	0.7613	0.6723	0.8663	0.8915	0.9934
GAT	0.8295	0.7971	0.8491	0.6484	0.8992	0.7103	0.7017	0.7734	0.5167	0.8623	0.7214	0.6576	0.8539	0.8805	0.9924
GAT(K=8)	0.8578	0.8427	0.8859	0.7453	0.9234	0.7237	0.7213	0.7969	0.5705	0.8792	0.7111	0.6564	0.8489	0.8764	0.9931
HAN	0.8687*	0.8548*	0.8942*	0.7640*	0.9294	0.7453	0.7325	0.8046*	0.5893	0.8842	0.7425	0.6681	0.8611	0.8877	0.9924
DHGAT$_{NP}$	0.8778	0.8640	0.8984	0.7730	0.9346	0.7844	0.7651	0.8245	0.6290	0.9048	0.8732	0.7562	0.9313	0.9427	**0.9961**
DHGAT	**0.8791**	**0.8654**	**0.8990**	**0.7736**	**0.9366**	**0.7883**	**0.7657**	**0.8252**	**0.6320**	**0.9071**	**0.8800**	**0.7581**	**0.9314**	**0.9435**	0.9959
Impv.	1.20%	1.23%	0.54%	1.26%	0.71%	4.84%	3.89%	2.56%	7.20%	2.15%	8.78%	6.99%	4.93%	3.99%	0.17%



(a) Normal Dataset (b) Hard Dataset (c) Long Tail Dataset

Figure 3: Parameter sensitivity of neighbor count.



(a) Normal Dataset (b) Hard Dataset (c) Long Tail Dataset

Figure 4: Parameter sensitivity of α.

number of transactions and **0.342%** on SPV (shop page view) compared to NCF. As for long-tail problem, we filter out those shops that have less than 10 exposures in the training set. On these shops DHGAT improves CTR (click through rate) by **48.1%** and CVR (conversion rate) by **72.8%** on the base of NCF.

6 RELATED WORK

Our work in this paper is closely related to the research of e-commerce search and graph neural networks.

E-commerce Search. E-commerce search is an important problem that has been widely studied both in academic and industrial community. Early studies mainly focus on how to support search for structured product information [6, 19]. Later, e-commerce search studies move to exploit search logs for improving searching performance [7, 22], especially for personalized search [1, 16]. There are also a variety of works on applying learning to rank techniques to e-commerce search for optimization of different retrieval metrics [14, 17]. Recently, research efforts have been attracted to latent embedding based e-commerce search models [2, 23, 31], which jointly learn distributional representations of users and products for better retrieval results.

Graph Neural Networks. Graph neural networks aim to generalize neural network models to graph structured data, such as convolution based GNN [18], attention based GNN [24] and sample and aggregation based GNN [12]. Lately researchers also deployed GNNs in recommender systems or e-commerce search to show consistent performance gains: leveraging GNNs to model user's interactions with items [28], to model session based temporal transitions [27], to model social homophily in user preference [10] and to model attributed multiplex heterogeneous network in e-commerce [3].

7 CONCLUSION AND FUTURE WORK

In this paper, we propose a dual heterogeneous graph attention network with transferring knowledge from product search for shop search in e-commerce. DHGAT is a deep and end to end framework that takes advantage of graph to enhance both ID representation and text representation. Specifically, it employs a hierarchical attention framework to compose homogeneous and heterogeneous neighbors simultaneously. Borrowing item neighbor title text from product search helps relieve the semantic gap between user queries and shop names. Besides, the proposed neighbor proximity loss provides strong additional guidance for learning graph topological structure. DHGAT further incorporates user features to better characterize user intention and retrieve personalized shop search results. We conduct extensive experiments in large scale offline evaluation and online A/B test. The results demonstrate the significant superiority of DHGAT over strong baselines and the effectiveness of the usage of heterogeneous graph.

For future work, we plan to investigate how to efficiently introduce user historical behavior of both shop search and product search. We also will explore more sophisticated relationships existing in user interaction heterogeneous graph and leverage them to enhance e-commerce search performance.

ACKNOWLEDGMENTS

This work was supported by Alibaba Group through Alibaba Innovative Research Program and National Natural Science Foundation of China (No. 61872278). We thank the anonymous reviewers for their valuable comments and suggestions. We also thank our colleagues: Honggang Wang, Lifeng Wang and Jiancai Liu for algorithm and engineering support on this work.

REFERENCES

[1] Qingyao Ai, Daniel N Hill, SVN Vishwanathan, and W Bruce Croft. 2019. A zero attention model for personalized product search. In *CIKM*. 379–388.
[2] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. 2017. Learning a hierarchical embedding model for personalized product search. In *SIGIR*. 645–654.
[3] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. *arXiv preprint arXiv:1905.01669* (2019).
[4] Liheng Chen, Yanru Qu, Zhenghui Wang, Lin Qiu, Weinan Zhang, Ken Chen, Shaodian Zhang, and Yong Yu. 2019. Sampled in Pairs and Driven by Text: A New Graph Embedding Framework. In *WWW*. 2644–2651.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[6] Huizhong Duan and ChengXiang Zhai. 2015. Mining coordinated intent representation for entity search and recommendation. In *CIKM*. 333–342.
[7] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. A probabilistic mixture model for mining and analyzing product search log. In *CIKM*. 2179–2188.
[8] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.
[9] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-Guided Heterogeneous Graph Neural Network for Intent Recommendation. In *KDD*. 2478–2486.
[10] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *WWW*. 417–426.
[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. 855–864.
[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[14] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *KDD*. 368–377.
[15] 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.
[16] Dietmar Jannach and Malte Ludewig. 2017. Investigating personalized search in e-commerce. In *The Thirtieth International Flairs Conference*.
[17] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2017. On application of learning to rank for e-commerce search. In *SIGIR*. 475–484.
[18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[19] Soon Chong Johnson Lim, Ying Liu, and Wing Bun Lee. 2010. Multi-facet product information search and retrieval using semantically annotated product family ontology. *Information Processing and Management* 46, 4 (2010), 479–493.
[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*. 3111–3119.
[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.
[22] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. 2018. User intent, behaviour, and perceived satisfaction in product search. In *WSDM*. 547–555.
[23] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning latent vector spaces for product search. In *CIKM*. 165–174.
[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[25] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *KDD*. 839–848.
[26] 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.
[27] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*. 346–353.
[28] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. 974–983.
[29] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR Meets Graph Embedding: A Ranking Model for Product Search. In *WWW*. 2390–2400.
[30] 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 *KDD*. 2347–2357.
[31] Kui Zhao, Yuechuan Li, Zhaoqian Shuai, and Cheng Yang. 2018. Learning and Transferring IDs Representation in E-commerce. In *KDD*. 1031–1039.

A APPENDIX

In the appendix, we firstly give the implementation details of our proposed model and compared methods to help in reproducibility. Then we describe more about the used dataset, including preprocessing procedures, graph neighbor construction and dataset statistics. Finally, we provide details of user features which contribute to characterizing customer intentions and retrieving personalized results.

A.1 Implementation Details of DHGAT

Minibatch Implementation. We adopt mini-batch training technique to calculate the gradients of Eq. (16). As described in Algorithm 1, for each iteration, we consider B triple sets $< u, q, s >$. For each pair $< q, s >$, the homogeneous and heterogeneous neighbors $\mathcal{N}_o(q)$, $\mathcal{N}_e(q)$, $\mathcal{N}_m(q)$, $\mathcal{N}_o(s)$, $\mathcal{N}_e(s)$, $\mathcal{N}_m(s)$ as well as the CTR signal y_{uqs} will be together input into the model for training. After one epoch is finished, the batches partition will be randomly reset to make the model more fully optimized. In each minibatch, since numbers of neighbors are different for queries and shops, we sample by weight a fix-sized set of neighbors of each node instead of using its full set neighbors. Note that if some node has too few neighbors, the sampled set may contain duplicates. In particular, if one node doesn't have any neighbors, we will instead use itself to fill the sampled set. To make computation more efficient, we leverage a negative sampling strategy during batch training when calculating neighbor proximity loss (See below for details).

Algorithm 1: Minibatch Implementation of DHGAT

```
Initialize model parameters Θ randomly;
for iteration in 1, 2, ... do
    Pick a minibatch of B triple sets < u, q, s > and the
      corresponding labels y_uqs;
    Sample by weight (B × 6N) neighbors N_o(q), N_e(q),
      N_m(q), N_o(s), N_e(s), N_m(s);
    Compute loss L_1(Θ) according to Eq. (14);
    Batch-in sample (B × N × 2N_neg) negatives for B pairs
      < q, s >;
    Compute loss L_2(Θ) according to Eq. (15);
    Compute gradients ∇L(Θ) according to Eq. (16);
    Update model: Θ = Θ − ϵ∇L(Θ);
end
```

Negative Sampling. To keep the computational pattern of each mini-batch fixed and more efficient, we make use of batch negative sampling technique when calculating the neighbor proximity loss. Specifically, take the query for example, suppose the batch size is B, that is, we have B pieces of queries. And we have to sample $(B \times N)$ homogeneous query neighbors within this batch. For each pair (q, q') of a piece of query q and its homogeneous neighbor $q' \in \mathcal{N}_o(q)$, we sample N_{neg} negatives from $((B-1) \times (N+1))$ queries, i.e., excluding query q itself and its homogeneous neighbors. After that we compute the mini-batch neighbor proximity loss according to Eq. (15).

Hyperparameters. The hyperparameter settings for DHGAT are as follows: (1) Embedding layers are randomly initialized with the default setting of TensorFlow xavier_initializer. For the other layers, weight parameters are initialized in the same way and bias parameters are assigned with TensorFlow zero_initializer. (2) The dimensions of identity embeddings and word embeddings are both set to 256. (3) The graph attention layer of our model is set to 1 in light of scalability and efficiency. (4) The mini-batch size is set to 512 and AdaGrad [8] is used as the optimizer, with initial learning rate is set to 0.015. (5) The number of two-tower layers L is set to 3 and hidden neurons of each layer are formulated as 512, 256, 128 from bottom to top, respectively. (6) LeakyReLU is adopted as activation function for graph attention layers and the negative slope coefficient alpha is set to 0.01. (7) The L2 regularization hyperparameter λ is set to 10^{-6}.

Hardware and Software. The proposed models are trained using Alibaba's distributed cloud platform which is based on TensorFlow 1.4 and Python 2.7. For the sake of training efficiency, we employ the batch neighbor sampling while training technology, which is supported by Behemoth, a large graph computation library used internally in Alibaba. In our experiments, the trained parameters are distributed on 40 workers (12 CPU cores for each worker) and updated asynchronously.

A.2 Implementation Details of Baselines

A.2.1 Classical Methods.

- NCF [13]. We re-implemented NCF according to the codes [3] provided by the corresponding author. Instead of inputting user and item embeddings, we feed query and shop identity embeddings into two-tower architecture and keep numbers of neurons of each layer unchanged for fair comparison.
- DSSM [15]. We adopt the latest NLP model BERT [5] as text feature extractor in DSSM setting. Note that due to characteristics of search matching, we compute inner product instead of cosine similarity as in [15]. The dimension of BERT embedding is set to 768. And parameters of BERT are pretrained using corpus in e-commerce.
- DNN. In this model, we feed into left tower concatenation of query identity embedding and BERT text embedding, right tower concatenation of shop identity embedding and BERT text embedding.

A.2.2 Two-stage Methods.

- GEPS [29]. We use DeepWalk [21] for pretraining graph embeddings in this two-stage method, which is implemented using Alibaba internal graph framework AliGraph. The dimension of node embedding is set to 512 and random walk length is set to 3. Note that DeepWalk is trained only with homogeneous relations, i.e., query to query and shop to shop.
- TGE-PS [4]. Different from GEPS, this method leverages pretrained text-driven graph embeddings. We implemented it with Bi-LSTMs over word embeddings and character embeddings as in original paper [4]. Dimension of node embeddings is same as DeepWalk in GEPS and half of it is the number of hidden neurons of Bi-LSTMs.

A.2.3 GNNs Methods.

[3] https://github.com/hexiangnan/neural_collaborative_filtering

Table 2: Basic Statistics of Dataset.

Dataset	# queries	# shops	# items	# users	# interactions
Train	$2.1 * 10^7$	$7.7 * 10^6$	$1.9 * 10^7$	$6.2 * 10^7$	$3.3 * 10^9$
Normal Test	$4.2 * 10^6$	$4.8 * 10^6$	-	$1.2 * 10^7$	$3.4 * 10^8$
Hard Test	$3.5 * 10^5$	$8.8 * 10^5$	-	$5.2 * 10^5$	$2.5 * 10^6$
Long Tail Test	$9.7 * 10^4$	$1.4 * 10^5$	-	$1.2 * 10^5$	$2.5 * 10^5$

Table 3: Statistics of Graph Relations.

Relation	# weighted edges	# edges
query2query	$7.4 * 10^9$	$1.6 * 10^{10}$
query2shop	$9.9 * 10^7$	$3.8 * 10^8$
query2item	$8.4 * 10^7$	$2.1 * 10^8$
shop2shop	$9.4 * 10^8$	$2.9 * 10^9$
shop2query	$9.9 * 10^7$	$3.8 * 10^8$
shop2item	$1.9 * 10^7$	$2.1 * 10^8$

- PinSAGE [28]. We re-implemented this method on the Alibaba distributed cloud platform based on the GraphSAGE [12] codes available on GitHub [4]. The dimensions of node embeddings and two-tower structure are kept same for fair comparison. The search space of number of sampling neighbors is $\{3, 4, 5, 6, 7, 8, 9, 10\}$. Note that this method doesn't incorporate text information.
- GAT [24]. The codes [5] released by the authors could not directly generalize to our dataset as they compute attention over all neighbors. We re-implemented with attention focused only on sampled neighbors for efficiency. The sampling number is searched within same space as PinSAGE.
- HAN [26]. We refered to the author's released codes [6] and re-implemented this method in a distributed way. In our setting, the aspects of semantic attention of queries and shops are both 2.

A.3 Datasets

A.3.1 Preprocessing.

As is mentioned before, when constucting homogeneous neighbors of queries, we need the session segment data. User behavior sequences are segmented into sessions based on time interval to prevent user intention drift. In our experiments, the time interval is set to 15 minutes. The other neighbor relations are constructed from shop search and product transaction logs collected from Taobao mobile App. We sum neighbor relation (i.e., graph edge) counts over all 9 training days to build the complete large graph. Shop IDs and item IDs are designated by Taobao system universally while query IDs are derived through hashing. For words segmentation of query texts, shop names and item titles, we adopt the Alibaba internal tool AliWS.

A.3.2 Statistics.

We report basic statistics of dataset and graph relations as in Table 2 and Table 3. Table 2 shows numbers of queries, shops, items, users and interactions in train set and test set (i.e., *Normal*, *Hard* and *Long Tail*). Note that items were not involved in test set since we cannot make use of transaction records dealed at test day. Table 3 shows numbers of neighbor relationships used during training. Recall that we assign weights to edges based on the co-occurrence times of nodes, in Table 3 we give numbers of weighted edges and non-weighted edges respectively.

[4]https://github.com/williamleif/GraphSAGE
[5]https://github.com/PetarV-/GAT
[6]https://github.com/Jhy1993/HAN

A.4 Details of User Features

We incorporate three groups of additional user features, which are described as follows:

- **User profile information.** This group primarily consists of user demographic features, which can help better understand user intentions and characterize user preferences in a fine-grained degree. Concretely, these features are gender, age, city, phone brand, etc. Some features (e.g., gender, city) are categorical features that could be vectorized directly via embedding look-up. And other features (e.g., age) are numerical features which will be discretized according to some thresholds and then embedded.
- **User consumption level.** This group of features mainly depicts the purchasing power and consumption habits of users, with which we can infer user's tastes and retrieve search results with high satisfaction. These features include vip type, vip score, phone price, city house price, etc. Vip level has 7 discrete feature states used in Taobao internally. Other numerical features (e.g., city house price) are collected or predicted from public markets.
- **User activity level.** The features of this group describe Taobao users activity at different granularities. For example, user's daily browsing time at Taobao mobile app, user's weekly login times, user's monthly pay amount, user's yearly luxury deals, etc. Activity level features could be utilized to capture user biases and special interests. Most of these features are continuous and need to be discretized with regard to selected threshold values.