Learning Better Representations for Neural Information Retrieval with Graph Information

Xiangsheng Li[1], Maarten de Rijke[2,3], Yiqun Liu[1*], Jiaxin Mao[1], Weizhi Ma[1], Min Zhang[1] and Shaoping Ma[1]

[1]Department of Computer Science and Technology, Institute for Artificial Intelligence, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China

[2]University of Amsterdam, Amsterdam, The Netherlands [3]Ahold Delhaize, Zaandam, The Netherlands

yiqunliu@tsinghua.edu.cn,m.derijke@uva.nl

ABSTRACT

Neural ranking models have recently gained much attention in information retrieval (IR) community and obtain good ranking performance. However, most of these retrieval models focus on capturing the textual matching signals between query and document but do not consider user behavior information that may be helpful for the retrieval task. Specifically, users' click and query reformulation behavior can be represented by a click-through bipartite graph and a session-flow graph, respectively. Such graph representations contain rich user behavior information and may help us better understand users' search intent beyond the textual information. In this study, we aim to incorporate this rich information encoded in these two graphs into existing neural ranking models.

We present two graph-based neural ranking models (*EmbRanker* and *AggRanker*) to enrich learned text representations with graph information that captures rich users' interaction behavior information. Experimental results on a large-scale publicly available benchmark dataset show that the two models outperform most existing neural ranking models that only consider textual information, which illustrates the effectiveness of integrating graph information with textual information. Further analyses show how graph information complements text matching signals and examine whether these two models can be adopted in practical applications.

KEYWORDS

Neural ranking; Network embedding; Graph neural network

ACM Reference Format:

Xiangsheng Li, Maarten de Rijke, Yiqun Liu, Jiaxin Mao, Weizhi Ma, Min Zhang, and Shaoping Ma. 2020. Learning Better Representations for Neural Information Retrieval with Graph Information. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411957

*Corresponding author.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00

https://doi.org/10.1145/3340531.3411957

1 INTRODUCTION

Relevance estimation is a central problem in information retrieval (IR) research, which aims to learn a scoring function to determine the degree of relevance of a document with respect to a query. Traditional solutions to this problem include both statistical models (e.g., BM25 [30]), and learning-to-rank models that combine multiple ranking functions [20]. Recently, neural ranking models have drawn attention for their ability to automatically extract features from raw text. They can be grouped into two categories, namely representation-based and interaction-based models [26]. Representation-based models integrate query and document information into vector representations while interaction-based models capture fine-grained interaction signals between query and document. These models aim to capture the degree to which there is a semantic match between query and document based on textual information. However, other kinds of information that are known to be informative are rarely being exploited by neural ranking models.

Besides the content information of search results, logged user behavior has shown great potential for search intent understanding and user behavior modeling has been extensively explored for improving relevance estimation [34]. To better exploit user behavior information, we can use graph structure to represent user behavior and further improve retrieval models. Previous studies have shown that incorporating such graph information can bring substantial improvements in many IR applications [1, 15, 21]. For example, Jiang et al. [15] propose to enrich vector representations of documents and queries by mining the click-through bipartite graph. A user's query intent is thus enriched with other adjacent interaction behavior. Figure 1 shows an example graph recording the interaction information of a web search session. A user first issued the

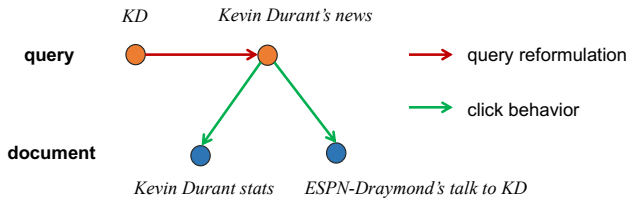


Figure 1: An example of graph information in web search. Orange arrows indicate query reformulations, green arrows link queries to documents being clicked in response to the query. This information can help to better understand a user's search intent for the first issued query "*KD*".

abbreviated query "*KD*" but did not find relevant documents. Then the query was reformulated to a clearer query "*Kevin Durant's news*" and it leads the user to probably useful documents. We can see that the search engine does not understand the user's intent behind the vague query "*KD*." If the interaction behavior information in the graph is adopted, it is possible for the search engine to help other users with the same query to obtain useful results more efficiently.

Despite the fact that graph-based information may be a rich source of knowledge beyond textual information, most existing neural ranking models do not take graph information into consideration. We propose to integrate graph information into a unified neural ranking model framework. We aim to learn better query/document representations with graph information. It is important to note that we take a representation-based ranking model as a basic ranking model rather than interaction-based ranking model because interaction-based models do not learn an independent representation for a specific query or document. Motivated by two main graph modeling methods [36] (network embeddings and graph neural networks), we propose two graph neural ranking models, namely an *embedding-based neural ranker* (EmbRanker) and an *aggregation-based neural ranker* (AggRanker). EmbRanker learns embeddings by leveraging the skip-gram model on a graph [22] to jointly preserve textual information and graph-based information. AggRanker learns embeddings by aggregating each local neighborhood from text content to the structure level.

While a wide range of information can be represented in different kinds of graphs, in this study, we focus on two graphs: the session-flow graph and the click-through bipartite graph. The session-flow graph describes how users reformulate queries, which can enrich the learned query embeddings with semantic information from other related queries in the same session context. The click-through bipartite graph specifies co-clicked documents with respect to a given query, which reduces the lexical gap between query and document embeddings. Compared to previous neural ranking models, the learned representations can better model users' search intent, which is particularly important for long-tail queries [15, 18].

We evaluate the proposed models, EmbRanker and AggRanker, on a large-scale publicly available test collection [4] from a commercial search engine, Sogou.com. Experimental results show that EmbRanker and AggRanker significantly outperform existing retrieval models. We also investigate graph parameters and computational costs to analyze the practical feasibility of incorporating graph information into neural ranking models.

In summary, our main contributions are as follows:

(1) We develop two graph-based neural retrieval models that utilize the session-flow graph and click-through bipartite graph to build better query and document representations for document retrieval. To the best of our knowledge, this is the first attempt to integrate graph-based information into neural ranking models.
(2) We conduct extensive experiments on a large-scale public test collection. Experimental results show that our proposed models significantly outperform most existing retrieval models, which illustrates the effectiveness of incorporating graph information into neural ranking models.
(3) We systematically analyze the effectiveness of using different graphs and the efficiency of two proposed models, thereby providing a solid understanding of how to effectively utilize graph-based information in the retrieval task.

2 RELATED WORK

2.1 Neural retrieval models

Existing retrieval models can be divided into two categories [12], namely *representation-based* and *interaction-based* models. Representation-based models aim to build a good semantic representation of queries and documents. DSSM [14] is the first successful representation-based model for retrieval tasks; it represents two input texts with a unified process by using a multi-layer perceptron (MLP) transformation. Convolutional networks [13] have been employed to replace MLPs to represent the input text. They are computationally efficient but lose fine-grained semantic information (e.g., passage or sentence-level relevance [19]). On the other hand, interaction-based models build local interactions between query and document, which capture more fine-grained semantic information. KNRM [37] utilizes kernel pooling to build multi-level soft matches between query and document. Matchpyramid [27] defines a symmetric interaction function to model term similarities between query and document.

2.2 Graph-based information retrieval

Graph-based information has been widely studied and exploited in the IR literature. Based on link structure, a series of ranking algorithms have been proposed (e.g., PageRank [25], HITS [17]). Previous studies also use different types of graphs to represent a user's behavioral information. Jiang et al. [15] enrich vector representations by using a click-through bipartite graph to reduce the lexical gap between query and document. However, this method is only based on BoW vectors and ignores low-frequency words, which leads to a loss of neighbor information. Click graphs have also been used to learn query intent [21] to help improve search performance, which illustrates that behavior graph is useful to improve intent modeling. Zhang et al. [42] learn query and item embeddings to approximate the external pre-trained graph embedding for product search. But this method requires external pre-training and its fixed graph embedding may be incompatible with a specific task.

To address the problems of existing graph-based ranking models [15, 42], we use neural networks to extract a dense representation that preserves the structural information in graphs and learn graph representations in an end-to-end manner. In our work, we utilize two different graphs, namely the session-flow graph and the click-through bipartite graph to improve neural ranking models. These graphs capture detailed interaction behavior information, which is helpful for modeling user search intent.

2.3 Graph-based deep learning

Recently, many efforts have been made to adapt deep neural networks to graph-structured data. The key idea is to enrich the representation with specific structural information and to preserve the node information. Existing graph modeling methods can be categorized into two classes: network embeddings and graph neural networks [36]. Network embeddings leverage the skip-grams method [22] and sample positive and negative samples to encode rich graph-structured relationships. For example, DeepWalk [28] and Node2vec [11] employ skip-gram techniques to learn structural

regularities present within random walks. Similar structures are also applied on text [31] and image [2] classification to manage supervised loss and structure loss. Specifically, their works only focus on classification task while our work focuses on text matching task. These models are computationally more expensive due to the use of sampling, however, their inference time is much lower than the training process. On the other hand, graph neural networks directly perform feature extraction in the graph domain by aggregating information from the neighbor nodes. Their computational procedures are usually the same during the training and testing process. Graph Convolutional Networks (GCNs) [8, 16] incorporate graph information into neural networks by generalizing traditional convolutional filters from images to graphs. Similar structures are also utilized in text classification [38] and recommender systems [39].

In our work, based on two graph modeling methods, we present two graph-based neural retrieval models, EmbRanker (based on the network embedding method) and AggRanker (based on graph neural networks), for document retrieval.

3 PRELIMINARIES

We aim to use graph information to build representations of queries and documents so as to improve basic ranking models that only exploit textual information. In this section, we give the notation and introduce the basic ranking models used in our framework.

3.1 Notation

Let $\mathcal{G} = (V, E)$ be an undirected graph consisting of a set of nodes V and a set of edges E. We focus on two graphs: the session-flow graph and the click-through bipartite graph, where V consists of queries $q \in Q$ and documents $d \in D$. The set E contains two kinds of edges $q \rightarrow q$ and $q \rightarrow d$, which represent consecutive queries in the same session and documents clicked following a query. Given a node with its text content $u = \{w_1, \ldots, w_u\} \in V$, we write $\mathcal{N}(u)$ for its neighborhood, i.e., the set of nodes connected to it. The graph $\mathcal{G}$ is generated from user behaviors on the training set. Given a query q, document d, and graph $\mathcal{G}$, the goal of our models is to estimate the relevance of d by generating high-quality representations that preserve both structural proximity information in the graph and semantic information in content.

3.2 Basic ranking models

We focus on enriching learned representations with graph information. Therefore, we build on representation-based ranking models as the basic ranking models instead of interaction-based ranking models since interaction-based models do not learn an independent representation for a specific query or document. The basic ranking model includes two parts: a text encoder and an output layer. To fairly compare the effectiveness of different text encoders, we apply the same output layer to predict relevance. Depending on the specific situations in practical systems, the basic ranking model could be any representation-based ranking models. In our work, we adopt three representation-based models as our basic ranking models, i.e., ARCI [13], DSSM [14] and Transformer [33].

3.2.1 ARCI. Given the textual content of a query or document, ARCI takes as input the embedding of its words and summarizes the meaning of that content through a layer of convolutions and pooling, until reaching a fixed length representation in the final layer. We write $\mathbf{x}_i \in \mathbb{R}^L$ to denote the word embedding with L dimensions. An n-gram in the text content is represented as the concatenation of words $\mathbf{x}_{i:j} = [\mathbf{x}_i \oplus \mathbf{x}_{i+1} \oplus \cdots \oplus \mathbf{x}_j]$. The final text representation is computed as follows:

$$\begin{aligned} c_{k,i} &= f(\mathbf{w}_k^T \mathbf{x}_{i:i+h-1} + b_k) \\ \hat{c}_k &= \max\{c_{k,1}, c_{k,2}, \ldots, c_{k,n-h+1}\} \\ \mathbf{c} &= [\hat{c}_1 \oplus \hat{c}_2 \oplus \cdots \oplus \hat{c}_K], \end{aligned} \tag{1}$$

where K is the number of convolution kernels, $\mathbf{w}_k \in \mathbb{R}^{hL}$ and $b_k \in \mathbb{R}$ are the weights in the k-th convolution kernel, extracting a window of h words to produce a local feature. Also, f is an activation function such as the ReLU. Then, the most important feature over all positions is captured by the max-pooling function. The features of different kernels are concatenated as the final text representation, denoted as $\mathbf{c}$.

3.2.2 DSSM. The first layer of the original DSSM model is a word-hashing layer, which transforms the high-dimensional term vector of the query/document to a low-dimensional letter-trigram vector. However, this layer is not applicable to our Chinese dataset since a Chinese sentence is composed of single words, not fine-grained letters. To address this problem, we replace the word-hashing layer as an Average Word Embedding (AWE) layer [41], which induces text embedding by averaging the embedding vectors of all terms. AWE has been shown to be effective in previous studies [41] and is able to learn good text representation.

In summary, in our framework DSSM first averages the word embedding of all terms in the text content, and then it applies multi-layer neural networks to map text features to a semantic representation. The final text representation is computed as follows:

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \tanh\left(W_0 \cdot \frac{\sum_i \mathbf{x}_i}{T} + b_0\right) \\ \hat{\mathbf{x}}_j &= \tanh\left(W_j \cdot \hat{\mathbf{x}}_{j-1} + b_j\right), j = 1, \ldots, J, \end{aligned} \tag{2}$$

where T is the text length and J is the number of hidden layers. The final $\hat{\mathbf{x}}_J$ is considered as the text representation.

3.2.3 Transformer. The Transformer [33] is a seq2seq model that relies on (self-)attention and several identical layer stacks. Each layer is composed of a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Multi-headed attention builds multiple "representation subspaces" and heads governed by separate sets of W_Q, W_K, W_V weight matrices, which are calculated as follows:

$$\begin{aligned} head_i &= \text{Attention}(HW_Q^i, HW_K^i, HW_V^i) \\ \text{Attention}(Q, K, V) &= \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \end{aligned} \tag{3}$$

where H is an embedding matrix for the bottom encoder, i.e., query or document embedding matrix. We use this stack of N Transformer blocks and the output embedding of the last block is considered as the query or document representation.

3.2.4 Output layer. We use the above methods to build a query representation $\mathbf{e}_q$ and document representation $\mathbf{e}_d$, separately. To fairly compare the quality of learned representations, we use the

same relevance scoring function $s(q, d)$ for all basic models, which is a linear transformation for the concatenation of $\mathbf{e}_q$ and $\mathbf{e}_d$:

$$s(q, d) = W_s \cdot [\mathbf{e}_q, \mathbf{e}_d] + b_s. \tag{4}$$

The ranking loss $\mathcal{L}_{rank}$ is defined as a pairwise ranking loss: document d^+ is ranked higher than d^- in terms of ground truth preferences, with respect to a query q;

$$\mathcal{L}_{rank} = \max(0, 1 - s(q, d^+) + s(q, d^-)). \tag{5}$$

4 GRAPH-BASED RANKING MODELS

In this section, we introduce two graph-based neural ranking models that improve basic (textual) ranking models with graph information, namely EmbRanker and AggRanker. The workflow of the two models is illustrated in Figure 2. Below, we describe their details.

4.1 Overview

The motivation of our framework comes from existing graph modeling methods, which can be categorized into two classes: network embedding methods and graph neural network methods [36]. Network embedding methods aim to preserve the space proximity of the learned node representation with skip-grams while graph neural network methods directly aggregate information from the neighbor nodes. We believe that both solutions may be employed to incorporate graph information into the ranking model. Therefore, we represent graph information based on space proximity and feature aggregation, with an *embedding-based neural ranker* (EmbRanker) and an *aggregation-based neural ranker* (AggRanker), respectively. Both methods aim to learn a good representation that preserves both textual and structural information so that search intents can be better understood. Hence, for neural retrieval models, we focus on improving representation-based models since interaction-based models aim to learn an interaction embedding and do not independently learn a specific representation for a query or document.

4.2 Embedding-based neural ranker

The core idea of the *embedding-based neural ranker* (EmbRanker) is to generate embeddings that preserve both structured proximity information and text information. Specifically, structured proximity means that a node is close to its local neighborhoods and away from other nodes in the embedding space. To learn structured proximity information, we learn the embeddings of query and document by using a skip-gram based network embedding method. Figure 2a illustrates the overall workflow of EmbRanker.

For each input query, document pair, EmbRanker first adopts its encoding architecture to encode text information as an embedding and then optimizes with both ranking loss and structural loss. To learn structured representations, we adopt skip-grams with negative sampling on random walk-based paths in the graph, as network embeddings benefit from random walk-based paths in reducing bias towards certain nodes [10, 32]. The random walk method generates its paths as follows. We select a node v_1 as the initial node and then circularly pick up one of its neighborhoods as the next node until the path length reaches a desired length. In our work, every node in the graph is used as the initial node, which means each sampling run contains $|V|$ paths. This guarantees that all the nodes are included in the sampled paths. The transition from a node v_i to v_j is governed by the transition probability $p(v_j \mid v_i) = 1/N$, where N is the number of neighborhoods of v_i. After obtaining random walk-based paths, we apply skip-grams with negative sampling on the paths to capture proximity information [22]. The neighborhoods in the path within a certain window size L are considered as positive samples for the central node while other nodes are randomly picked as negative samples. We also incorporate a percentage ϵ of non-clicked documents as negative samples for query nodes. Then, we maximize the likelihood of positive samples S^+ and minimize the likelihood of negative samples S^-. The structural loss $\mathcal{L}_s$ is thus the overall negative log-likelihood:

$$\begin{aligned}\mathcal{L}_s &= -\mathcal{L}(S^+, S^- \mid \Theta) \\ &= -\sum_{(i,j)\in S^+} \log \sigma(\mathbf{e}_i^T \mathbf{e}_j) - \sum_{(i,k)\in S^-} \log \sigma(-\mathbf{e}_i^T \mathbf{e}_k),\end{aligned} \tag{6}$$

where $\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k$ denote embeddings of the central node, the positive sample and the negative sample, respectively. Θ are the model parameters; σ is the sigmoid function; the negative likelihood function is also called as structural loss, aimed at learning structured information present in the graph. The final loss function of EmbRanker $\mathcal{L}_{emb}$ consists of structural loss and ranking loss:

$$\mathcal{L}_{emb} = \mathcal{L}_{rank} + \lambda_s \cdot \mathcal{L}_s, \tag{7}$$

where λ_s is a trade-off parameter for structure loss $\mathcal{L}_s$. During the training process, the encoding architecture of a basic ranking model is optimized by $\mathcal{L}_{emb}$ such that the learned embedding preserves both textual and structural information.

4.3 Aggregation-based neural ranker

Compared to the EmbRanker, which uses skip-grams to learn embeddings in a pairwise manner, the *aggregation-based neural ranker* (AggRanker) learns graph information by directly propagating the information of all the neighborhoods to the focal node by a graph neural network (GNN). The key idea is to learn a neighborhood embedding that represents the local structure information up to a given depth K. Figure 2b illustrates the workflow of AggRanker, which is a two-stage encoding process.

In the first stage, AggRanker first collects all the neighborhoods of the focal node (query node or document node) within a given depth K, e.g., $(q_1, q_2, q_3, q_4, d_1, d_2, d_3)$ in Figure 2b. Then it uses the text encoder of a basic ranking model to map the content of each node into a textual semantic embedding as detailed in Section 3.2. We use $\mathbf{e}_v$ to denote the textual semantic embedding of the node v.

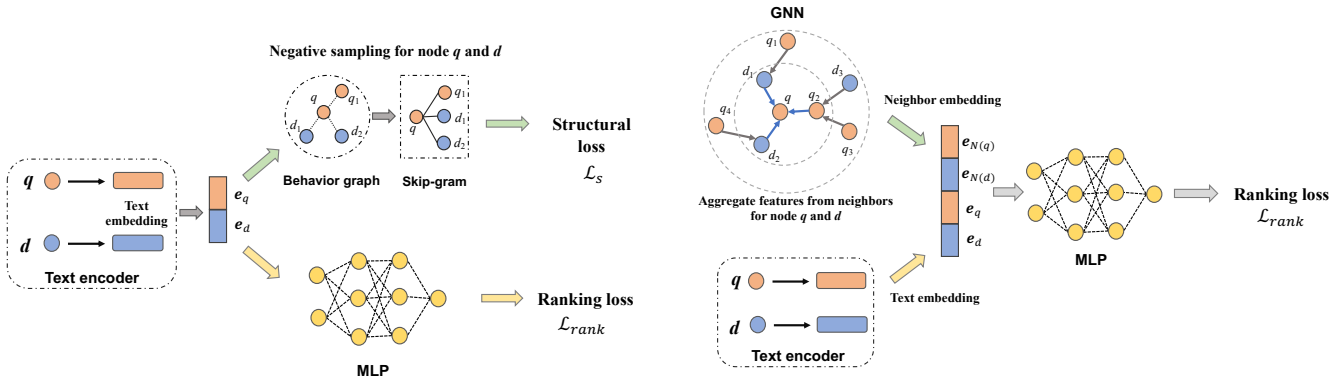
Second, AggRanker recursively aggregates the semantic embeddings of the neighborhoods layer by layer. For node v_i in the k-th layer, the aggregated embedding $\mathbf{h}_{v_i}^k$ is represented as:

$$\mathbf{h}_{v_i}^{k-1} = \sum_{v_j \in \mathcal{N}(v_i)\cup\{v_i\}} \tilde{m}_{ij} \mathbf{h}_{v_j}^{k-1} \tag{8}$$

$$\mathbf{h}_{v_i}^{k} = \tanh(W^k \cdot \mathbf{h}_{v_i}^{k-1} + b^k), \tag{9}$$

where the first aggregated embedding is the textual semantic embedding $\mathbf{h}_{v_i}^0 = \mathbf{e}_{v_i}$ and $\tilde{m}_{ij} \in \tilde{M}$ is the normalized symmetric adjacency matrix specifying the connection between v_i and v_j. Following the multi-layer Graph Convolutional Network (GCN) model [16], the matrix $\tilde{M}$ is given by

$$\begin{aligned}\tilde{M} &= \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \\ \tilde{A} &= A + I_N,\end{aligned} \tag{10}$$



(a) Embedding based neural ranker (EmbRanker).

(b) Aggregation based neural ranker (AggRanker).

Figure 2: Workflows of two graph-based neural retrieval models. EmbRanker (a) uses skip-grams with negative sampling to embed graph-based information for ranking. AggRanker (b) uses GNNs to encode graph-based information as neighbor embeddings and combines it with the original text embedding for ranking.

where $\tilde{A} \in \mathbb{R}^{|V|\times|V|}$ is the adjacency matrix with added self-connections, I_N is the identity matrix and $\tilde{D} = \sum_j \tilde{A}_{ij}$ is the degree matrix. In each layer, the network not only gathers the information from the previous layer, but also diffuses the current node information to the next layer. The number of layers K allows message passing among nodes that are at most K steps away. Global information can be learned in this information propagation manner.

We take the concatenation of the neighbor embedding $\mathbf{e}_{N(v_i)} = \mathbf{h}_{v_i}^K$ and textual embedding $\mathbf{e}_{v_i}$ as the final representation of the query-document pair. Similar to EmbRanker, we then apply a linear transformation to construct the objective function in Eq. 4 and learn using a pairwise ranking loss as in Eq. 5.

4.4 Discussion

EmbRanker and AggRanker are designed based on two important graph modeling methods [36]: network embeddings and graph neural networks. They share two common aspects. (1) EmbRanker and AggRanker are proposed for a representation-based model since we aim to enrich representation with both textual semantic and structural information. However, interaction-based models cannot independently learn a specific representation for a query or document, they are thus not considered in our framework. Experimental results show that by incorporating graph information into representation-based models, they can perform better than interaction-based models. We leave the combination of interaction-based models and graph information as future work. (2) Importantly, the basic ranking model is suggested to be computationally efficient since the text content of neighborhoods in the graph is also modeled. When using a large amount of neighborhoods, this brings computational challenge to the ranking system. Therefore, in practical systems, we suggest to optimize the computationally expensive part or reduce the number of parameters. We will discuss the computational costs of the two methods in Section 6.4.

There are important differences in both the training and test processes for EmbRanker and AggRanker. For training, we observe that EmbRanker requires additional preprocessing for graph traversals to sample positive and negative pairs while AggRanker directly models structural information during the training process. Preprocessing comes with a computational overhead. However, for testing, EmbRanker does not require the input of structural information while AggRanker still needs to consider graph information as input. This means that the inference time of EmbRanker is substantially shorter than for AggRanker, which is important in practice. See Section 6.4.

5 EXPERIMENTAL SETUP

5.1 Dataset

We evaluate the proposed methods on a large-scale, publicly available query log from a Chinese commercial search engine, Sogou.com, namely Tiangong-ST[1] [4]. Table 1 shows the statistics of our dataset. Tiangong-ST provides web search session data extracted from an 18-day search log. It contains weak relevance labels (i.e., click relevance labels [37]) derived by six different click models for all query-document pairs and human relevance labels for documents in the last query of 2,000 sampled sessions. Due to the context difference, the number of unique queries is smaller than the number of sessions. We use the 2,000 last queries as our test set for an ad-hoc retrieval task. Since the text content of neighborhoods is also considered in the graph modeling framework, we use document titles in both training and testing instead of the full document content, which ensures efficiency when incorporating multiple hops of neighborhoods. A similar experimental setup has previously been used; see, e.g., [37]. Prior studies [6, 37] have shown that weak relevance labels derived from click models can be used to train and evaluate retrieval models. Since the Partially Sequential Click Model (PSCM) [35] achieves the best relevance estimation performance among the six click model alternatives, we employ click labels from the PSCM for training and validation. For testing, we use the provided five-graded human annotated relevance labels to evaluate ranking performance.

Graph-structured information is extracted from consecutive queries in a session (session-flow graph) and clicked documents (click-through bipartite graph). We build graphs only based on user behavior in the training set. A total of 87.1% of the queries and 54.5% of the documents in the test set can be found in the training set. For the validation set, 94.0% of the queries and 57.1% of the

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

Table 1: Statistics of the Tiangong-ST dataset and the generated graph. "All" means both the query click-through bipartite graph ($q-d$) and the session-flow graph ($q-q$).

	Train	Valid	Test
#unique queries	39,777	1,111	610
#queries	344,942	4,888	2,000
#sessions	143,155	2,000	2,000
#avg. session length	2.41	2.44	3.21
#avg. click[1] per query	3.29	3.36	3.52
#avg. doc per query	9.60	9.58	9.59
	All	$q-d$	$q-q$
#nodes	92,926	39,776	86,109
#edges	145,445	72,050	73,395
#avg neighbors	3.012	3.347	1.705

documents are in the training set. For new queries or documents that are not in the graphs, EmbRanker is able to handle them since it does not need structural information in testing and AggRanker takes a fixed *NULL* embedding as the neighbor embedding. Previous studies have revealed several types of bias in SERPs such as "position" [7] (user's decaying attention in vertical direction), "trust" [40] (user's judgement is affected by website reputation) and "presentation" [35] (search results' display styles influence user's attention) factors. To reduce the possible bias in a SERP and obtain an unbiased estimation of result relevance, we use PSCM labels to generate pseudo clicked documents instead of direct clicks, where documents with a click probability over a fixed threshold are considered as having been clicked.

Following the setting in [37], we map click relevance scores into five-graded relevance grades in terms of the label distribution in the TREC 2014 session track [3] for validation, where documents with the top 29% click probability (grade 1 and higher) are considered relevant and being clicked.

5.2 Experimental settings

5.2.1 Baselines. Our baselines include four types of retrieval model: a probabilistic ranking model (i.e., BM25), neural ranking models, graph-based ranking models, and a pre-trained language model (BERT). The first two categories consider only text information (T) while the third considers both text and graph information (T+G):

- **BM25** (T): A popular probabilistic model-based ranking function proposed by Robertson et al [30].
- **Matchpyramid** [27] (T): A neural ranking model that first builds a word-level relevance interaction matrix and then applies CNNs to aggregate it into the final relevance estimation. We use a one-layer CNN with 64 (1×3) kernels and a (2×2) pooling size.
- **KNRM** [37] (T): A neural ranking model that uses a kernel pooling strategy to model multi-level semantic matching signals. We use 11 kernels as the default setting in the original paper (10 soft-matching and 1 exact-matching kernels).
- **ARCII** [13] (T): A neural ranking model that maps the word embeddings of query and document to an aggregated embedding by a CNN. we use a two-layer CNN, where the size of kernels and pooling in both layers are set to (3×3) and (2×2). There are 16/32 kernels in two layers.
- **ARCI** [13] (T): ARCI is a representation-based model as discussed in Section 3.2. It is used as a basic ranking model. We use a three-layer CNN where the filter windows sizes are 1 to 3 and there are 64 feature maps for each filter.
- **DSSM** [14] (T): DSSM is also used as a basic ranking model as discussed in Section 3.2. We replace the word-hashing layer as an Average Word Embedding (AWE) layer [41] to model Chinese sentences as discussed in Section 3.2. We use a three-layer DNN as in the original paper of DSSM; the hidden number of each layer is set to 50.
- **Transformer** [33] (T): Transformer is also used as a basic ranking model discussed in Section 3.2. We use a stack of 2 Transformer blocks with hidden size = 64, number of attention heads = 5. The number of parameters is reduced to some extent for the computational efficiency in the graph modeling.
- **VPCG** [15] (T+G): A graph-based model that enriches vector representations by using click-through bipartite graph to reduce the lexical gap between query and document. We use the default parameter settings $K = 20$ as in the original paper. Compared with our models, this method is only based on term frequency information and ignores low-frequency words, which loss a part of neighborhood information.
- **GEPS** [42] (T+G): A graph-based model that utilizes graph embeddings from an external unsupervised graph model to guide the representation learning for product search. We set the dimension of the graph embedding as 128. Compared with our models, this method only uses the fixed pre-trained graph embedding which may be incompatible with a specific task.
- **BERT** [9] (Pretraining+T): Pretrained language model BERT that has been shown to be very effective on the document ranking task [29]. We use the pretrained 12-layer *bert-base-chinese*[3] model and finetune it on our dataset.

Table 2: Ranking performance when using graph information on different basic ranking models. †, ‡, § indicates significant improvements with respect to the corresponding basic ranking models, respectively. (p-value ≤ 0.05).

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10
ARCI	0.6045	0.6628	0.7043	0.8257
+EmbRanker	0.6258†	0.6772†	0.7125†	0.8412†
+AggRanker	**0.6595**†	**0.6797**†	**0.7132**†	**0.8465**†
DSSM	0.6141	0.6550	0.7016	0.8253
+EmbRanker	0.6306‡	0.6668‡	0.7127‡	0.8408‡
+AggRanker	0.6401‡	0.6633‡	0.7104‡	0.8420‡
Transformer	0.5776	0.6378	0.6905	0.8263
+EmbRanker	0.6175§	0.6569§	0.6953	0.8348§
+AggRanker	0.6234§	0.6588§	0.6995§	0.8351§

5.2.2 Parameter settings. We implement our models using Pytorch. The parameters are optimized by Adam, with a batch size of 80 and a learning rate λ of 0.001. The dimension of the word embeddings is 50 and they are pretrained on a Chinese Wikipedia dataset[4] by

[1] Pseudo click drawn from PSCM labels, where documents with top 29% click probability (grade 1 or higher) are considered being clicked.

[3] https://github.com/google-research/bert/blob/master/multilingual.md

[4] http://download.wikipedia.com/zhwiki.

Table 3: Ranking performance of different retrieval models. † indicate statistically significant improvements over all the baselines except BERT. 1, 2 indicates a significant improvement over EmbRanker and AggRanker, respectively (p-value ≤ 0.05).

Model type	Feature	Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10
Probabilistic	Text	BM25	0.4528	0.5533	0.6244	0.7856
Neural ranking model	Text	MatchPyramid	0.4807	0.5565	0.6135	0.7849
		KNRM	0.5676	0.6172	0.6738	0.8176
		ARCII	0.5822	0.6363	0.6889	0.8285
		ARCI	0.6045	0.6628	0.7043	0.8257
		DSSM	0.6141	0.6550	0.7016	0.8253
		Transformer	0.5776	0.6378	0.6905	0.8263
Graph-based ranking model	Text+Graph	VPCG	0.5729	0.6369	0.6904	0.8242
		GEPS	0.5940	0.6573	0.6987	0.8347
		EmbRanker(ARCI)	0.6258†	0.6772†	0.7125†	0.8412†
		AggRanker(ARCI)	**0.6595**†	0.6797†	0.7132†	0.8465†
Pre-trained language model	Externel corpus + Text	BERT	0.6388[1]	**0.6926**[1,2]	**0.7277**[1,2]	**0.8503**[1,2]

using word2vec. The output dimension of the text representation is 50 for three different text encoders. For EmbRanker, the percentage ϵ of non-clicked documents in negative sampling is 0.4. Since the number of pairs in $\mathcal{L}_s$ is larger than a single pair in $\mathcal{L}_{rank}$, the trade-off parameter of the structure loss λ_s is set smaller, selecting from [0.1, 0.05, 0.01]. To model the multi-order connectivity in the graph, we set the window size of negative sampling for EmbRanker and the depth of aggregation layer for AggRanker as 3 and 2, respectively. We also report the effect of other depths in Section 6.3.2. Early stopping with a patience of 5 epochs is adopted during the training process. All experiments are conducted on a single NVIDIA GeForce GTX TITAN X GPU with 12GB-memory. We use NDCG (Normalized Discounted Cumulative Gain) as evaluation metric. The source code is publicly available[5].

6 RESULTS AND ANALYSIS

6.1 RQ1: Can our framework improve the representation-based ranking models?

Based on the results in Table 2 we arrive at the following lessons:

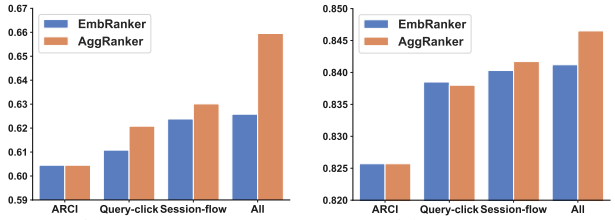
(1) We find that both of the proposed graph ranking models, EmbRanker and AggRanker, obtain significant improvements over the underlying basic ranking models, which suggests that users' behavior graph is helpful for improving the ranking performance and our frameworks can effectively exploit this information to learn better representations for the retrieval task.
(2) We find that AggRanker with the basic ranking model ARCI achieves the best ranking performance. It illustrates that modeling graph information by an aggregation function leads to a better graph representation than the skip-gram method. The aggregation function learns graph information in a separate graph embedding, thus it can better convey behavior information than the integrated embedding in skip-grams method.
(3) The Transformer does not perform as well as ARCI and DSSM. This is because it is not designed as an independent text encoder in retrieval tasks [33]. Regular usage of the Transformer is as an interaction-based model and to take the concatenation query and document with a [*SEP*] as input [29]. Similar findings were reported in [29]. Despite this, when using Transformer as the basic ranking model, our framework can also improve its ranking performance with graph information.

6.2 RQ2: How does our framework perform compared to existing ranking models?

In additional to traditional BM25, neural ranking models and graph-based ranking models, we also compare our methods with the state-of-the-art pretrained language model BERT [9]. See Table 3. To reduce overlap with the results in Table 2, we only list the results of our methods based on ARCI, the best-performing basic ranking model. From the table, we have the following lessons:

(1) The traditional probabilistic model BM25 performs worst compared with other retrieval models. This indicates that merely considering exact term matching is not enough to model the relevance between query and document on our collection.
(2) While some negative results were reported for representation-based models in ad-hoc retrieval tasks [24], we observe that most of representation-based models (ARCI, DSSM) can achieve better results than interaction-based models (MatchPyramid, KNRM, ARCII). Previous studies have shown that representation-based models can perform better than interaction-based models over short and popular queries [23]. In our dataset, 69.9% of the queries have a length shorter than 4 words and 87.1% of the test queries are observed in the training set, which means that the majority of the queries are short and popular queries. This phenomenon results in the better performance of representation-based models in our dataset.
(3) Considering the comparisons with text-based ranking models, we find that EmbRanker and AggRanker with ARCI outperforms all neural ranking models that only use text features. This confirms that graph information is useful for the retrieval task and well exploited in our frameworks.
(4) We also compare our methods with graph ranking models. Both proposed models outperform VPCG and GEPS significantly. As discussed in Section 2.2, VPCG suffers from losing neighborhood information while GEPS only uses an external fixed graph embedding to guide the training, which may be incompatible with a specific task. Our models overcome these shortcomings

[5] https://github.com/lixsh6/GraRetrieval-CIKM2020.



(a) NDCG@1 **(b) NDCG@10**

Figure 3: Ranking performance when using different graph informations. "All" means using query click-through bipartite graph ($q-d$) and session-flow graph ($q-q$) together.

by integrating graph information into a dense representation and learning in an end-to-end manner. From a retrieval point of view, our models provide more informative textual and graph information than VPCG and GEPS.

(5) For the pre-trained language model BERT, we find that it achieves the best performance on most evaluation metrics. This is probably because: (a) BERT has a much larger parameter size (110M parameters) than other ranking models and is pre-trained on an external corpus; (b) with a large model capacity, BERT can effectively model the interaction between the query and document [33]. Our models perform best among all the baselines that do not rely on external corpus and a resource-consuming pre-training process. As an aside, because it is preferable to use BERT as an interaction-based ranking model [29], the question of how to incorporate graph information into BERT is beyond the scope of this paper and left as future work.

6.3 RQ3: How does graph information contribute to the ranking performance?

In this section, we aim to understand the influence of different graph parameters (graph types, depth and size) on ranking performance. Due to space limitations, we only report NDCG@{1, 10} of ARCI as other results are qualitatively similar.

6.3.1 Effect of different graphs. In our work, we exploit two types of graph to build graph-structured information. To study which type of graph information contributes more to the overall ranking performance, we conduct an ablation study for the proposed models. In Figure 3, "Query-click" represents the click-through bipartite graph ($q-d$ edges) and "Session-flow" represents the session-flow graph ($q-q$ edges). We obtain the following lessons:

(1) Both types of graph information improve the original ranking model ARCI. This demonstrates the effectiveness of our proposed graph-based ranking model framework and shows that the two kinds of graph information provide useful features to enrich the representation of query and document.

(2) Comparing the two types of graph information, the models based on the session-flow graph achieve better performance than those based on the click-through bipartite graph. We explain this as follows: (a) the information of $q-d$ edges might be duplicated partially with information learned through the ranking loss; and (b) there exist more noisy signals in the click--through bipartite graph due to users' click noise [5].

(3) Between the two graph-based ranking models, AggRanker outperforms EmbRanker in most of cases. When combining two types of graph information, EmbRanker and AggRanker can both achieve better ranking performance than with a single source of graph information.

Table 4: Ranking performance of different graph depths, i.e., the window size L in EmbRanker and the propagation layer number K in AggRanker.

	EmbRanker		AggRanker	
Depth	NDCG@1	NDCG@10	NDCG@1	NDCG@10
0	0.6045	0.8257	0.6045	0.8257
1	0.6129	0.8369	0.6471	0.8418
2	0.6229	0.8373	**0.6595**	**0.8465**
3	**0.6258**	**0.8412**	0.6505	0.8451
4	0.6084	0.8377	0.6318	0.8380
5	0.6152	0.8370	0.6188	0.8382

Table 5: Ranking performance when using different graph sizes. Averaged results over 5 times sampling are reported. ▲ represents a significant improvement compared to the original model (i.e., 0%) with p-value $\leq$ 0.05.

	EmbRanker		AggRanker	
Percentage	NDCG@1	NDCG@10	NDCG@1	NDCG@10
0%	0.6045	0.8257	0.6045	0.8257
20%	0.5759	0.8294	0.6027	0.8282
40%	0.5912	0.8334	0.6265▲	0.8284
60%	0.6020	0.8350▲	0.6208▲	0.8391▲
80%	0.6177▲	**0.8415**▲	0.6330▲	0.8446▲
100%	**0.6258**▲	0.8412▲	**0.6595**▲	**0.8465**▲

6.3.2 Effect of depth in the graph. Proximity information of a central node in a graph enriches its textual semantic representation. However, at what distance in a graph do nodes still provide information that helps to enrich the representation? We tune the sampled window size L in EmbRanker and the propagation layer number K in AggRanker to study this problem. For simplicity, we use depth to represent window size and propagation layer number directly. The results in Table 4 provide the following lessons:

(1) The overall tendency of our proposed methods is similar, where the ranking performance first increases and then decreases as the graph depth increases. This illustrates that there exists a tradeoff between the improvement and depth. Remote neighborhoods bring more noise and thus have a negative impact on the ranking performance.

(2) The depth with the best performance for EmbRanker and AggRanker is around 3 and 2. This result indicates that neighborhoods within this depth are semantically relevant to their central nodes and other, more remote nodes are mostly noise.

6.3.3 Effect of graph size. Graph information can be considered as a type of side information for ranking models. To study the influence of the graph size, we vary the percentage of graph nodes by random sampling. We finally report the averaged results over sampling 5 times for each percentage in Table 5 and have the following lessons:

(1) The two proposed models do not perform well and even get worse when only using a small amount of graph information. This is because the models may easily suffer from overfitting with small graph sizes and the learned graph information influence the quality of the whole learned embedding.

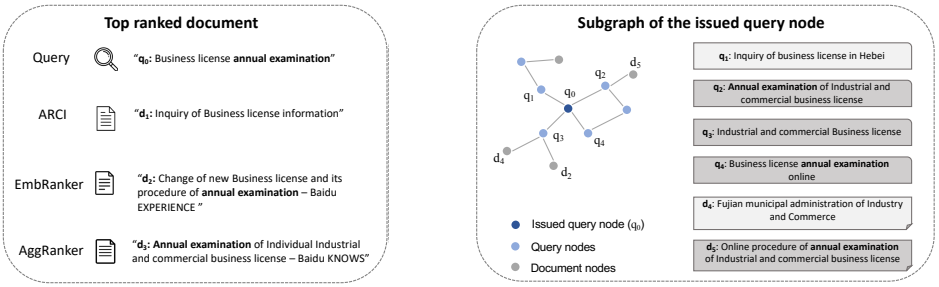


Figure 4: An example of top-ranked documents from different ranking models. Important words are in bold face. In the subgraph, q_1 and d_4 are irrelevant nodes with lighter background color.

(2) As the graph size gets larger, both models perform better and significantly outperform the original model (without using graph information). The minimal required graph size for a significant difference of the two models are 80% and 60%, respectively. This illustrates that the proposed models require a large volume of graph information to be effective.

6.3.4 Case study. To better understand what can be learned by the proposed models, we conduct a case study to analyze the relationship between the top-ranked document and the applied graph information. An example is shown in Figure 4, where the subgraph of the issued query q_0 is shown on the right. The result is translated from Chinese. We have the following observations:

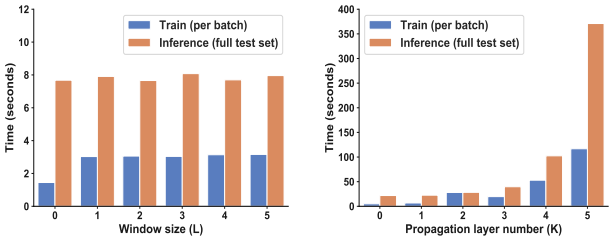
(1) The original model ARCI improperly provides a link to check the license information, which is irrelevant to the search intent of the issued query.
(2) The proposed graph-based models can leverage the graph to better infer user search intent and provide the desired information about annual examination. Importantly, the proposed models can discover the important words *annual examination* since the majority of the neighborhood nodes provide similar information on this search target.
(3) We find that the proposed models works well despite the presence of noisy nodes in the graph (e.g., q_0 and d_4) because the neighborhood information is aggregated over the majority of neighborhoods. This illustrates the robustness of the proposed models when using a large-scale graph information.

6.4 RQ4: How efficient is our framework?

Compared to traditional neural ranking models, our methods have more computational costs because of the graph information that they incorporate. For both proposed models, the efficiency of text encoder plays an important role since the text content of neighborhoods is also considered. This is why we suggest to use computationally efficient representation-based models. Specifically, for EmbRanker, its additional computational costs are only incurred during training since it does not require structural information during testing. For AggRanker, its computational costs increase as the depth of the graph information considered becomes larger.

Figure 5 summarizes the training and testing run-times for the proposed models in different depths. The basic ranking model is ARCI. Depth 0 indicates the ARCI without using graph information. We have the following observations:

(1) Compared with the original model (i.e., depth 0), EmbRanker doubles the run-time when using graph information. And there is no significant change at different depths (1 to 5) for both training and testing.



(a) EmbRanker **(b) AggRanker**

Figure 5: Run-times in different graph depths.

(2) For AggRanker, we find that its run-time substantially increases as the number of layers grows. At depth 1, its computational costs increase by about 33% and 3% in training and testing, respectively. For the best depth parameter 3, its inference run-time increases about 81%, however, its ranking performance NDCG@1 enhances 9%, which is a considerable tradeoff in practical ranking scenarios. In particular, the growth rate is mainly influenced by the neighborhood number N.
(3) We see that the computational costs of AggRanker are substantially higher than of EmbRanker and that AggRanker is more sensitive to variation in depth. Although EmbRanker does not perform as well as AggRanker, it benefits due to lower computational costs.

7 CONCLUSION

This paper proposes two graph-based neural ranking models for web search to enrich learned text representations with graph information that captures rich behavior information. Specifically, the proposals are based on a network embedding method (EmbRanker) and a graph neural network method (AggRanker), respectively.

Extensive experiments demonstrate the effectiveness of incorporating graph information into neural ranking models. Our models perform best among all the baselines that do not rely on external corpus or a time-consuming pre-training process. We further investigate how different graph parameters (i.e., graph types, depths, and size) influence ranking performance and the computational efficiency. Our work systematically analyzes the importance of adding graph information to a neural ranking model and provides a better understanding of how to model graph information effectively using neural architectures for IR.

In future work, we plan to incorporate graph information into interaction-based ranking models such as BERT. Our work only focuses on representation-based models; how to properly use graph information with interaction-based models remains an open problem. We believe that graph information is an important feature beyond traditional text matching techniques for neural ranking models and may help us build better web search systems.

ACKNOWLEDGEMENTS

This work is supported by the National Key Research and Development Program of China (2018YFC0831700), Natural Science Foundation of China (Grant No. 61732008, 61532011, 61902209), Beijing Academy of Artificial Intelligence (BAAI) and Tsinghua University Guoqiang Research Institute. This project is also funded by China Postdoctoral Science Foundation and Dr Weizhi Ma has been supported by Shuimu Tsinghua Scholar Program. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. 543–551.

[2] Thang D Bui, Sujith Ravi, and Vivek Ramavajjala. 2018. Neural Graph Learning: Training Neural Networks Using Graphs. In *Proceedings of the 2018 international conference on web search and data mining*. 64–71.

[3] Ben Carterette, Paul Clough, Mark Hall, Evangelos Kanoulas, and Mark Sanderson. 2016. Evaluating retrieval over sessions: The TREC session track 2011-2014. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 685–688.

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

[5] Weizhu Chen, Dong Wang, Yuchen Zhang, Zheng Chen, Adish Singla, and Qiang Yang. 2012. A noise-aware click model for web search. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 313–322.

[6] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. San Rafael: Morgan and Claypool.

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

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.

[9] 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).

[10] Yuxiao Dong, Jing Zhang, Jie Tang, Nitesh V Chawla, and Bai Wang. 2015. Coupledlp: Link prediction in coupled networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

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

[12] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2019. A Deep Look into Neural Ranking Models for Information Retrieval. *arXiv preprint arXiv:1903.06902* (2019).

[13] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.

[14] 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 *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2333–2338.

[15] Shan Jiang, Yuening Hu, Changsung Kang, Timothy Daly, Dawei Yin, Yi Chang, and Chengxiang Zhai. 2016. Learning Query and Document Relevance from a Web-scale Click Graph. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 185–194.

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

[17] Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46, 5 (1999), 604–632.

[18] Xiangsheng Li, Yiqun Liu, Xin Li, Cheng Luo, Jian-Yun Nie, Min Zhang, and Shaoping Ma. 2018. Hierarchical Attention Network for Context-Aware Query Suggestion. In *Asia Information Retrieval Symposium*. Springer, 173–186.

[19] Xiangsheng Li, Jiaxin Mao, Chao Wang, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. Teach Machine How to Read: Reading Behavior Inspired Relevance Estimation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 795–804.

[20] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[21] Hao Ma, Haixuan Yang, Irwin King, and Michael R Lyu. 2008. Learning latent semantic relations from clickthrough data for query suggestion. In *Proceedings of the 17th ACM conference on Information and knowledge management*.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

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

[24] Yifan Nie, Yanling Li, and Jian-Yun Nie. 2018. Empirical study of multi-level convolution models for ir based on representations and interactions. In *Proceedings of the 2018 ACM ICTIR*.

[25] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[26] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2017. A Deep Investigation of Deep IR Models. *arXiv preprint arXiv:1707.07700* (2017).

[27] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Thirtieth AAAI Conference*.

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

[29] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531* (2019).

[30] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*.

[31] Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A general framework for content-enhanced network representation learning. *arXiv preprint arXiv:1610.02906* (2016).

[32] Yizhou Sun, Brandon Norick, Jiawei Han, Xifeng Yan, Philip S Yu, and Xiao Yu. 2013. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7, 3 (2013), 11.

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

[34] Chao Wang, Yiqun Liu, and Shaoping Ma. 2016. Building a click model: From idea to practice. *CAAI Transactions on Intelligence Technology* 1, 4 (2016), 313–322.

[35] Chao Wang, Yiqun Liu, Meng Wang, Ke Zhou, Jian-yun Nie, and Shaoping Ma. 2015. Incorporating non-sequential behavior into click models. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 283–292.

[36] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).

[37] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conferenc*. ACM, 55–64.

[38] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference*, Vol. 33. 7370–7377.

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

[40] Yisong Yue, Rajan Patel, and Hein Roehrig. 2010. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th international conference on World wide web*. 1011–1018.

[41] Hamed Zamani and W Bruce Croft. 2016. Estimating embedding vectors for queries. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*. 123–132.

[42] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR Meets Graph Embedding: A Ranking Model for Product Search. *the web conference* (2019), 2390–2400.