CDSM: Cascaded Deep Semantic Matching on Textual Graphs Leveraging Ad-hoc Neighbor Selection

JING YAO, Microsoft Research Asia
ZHENG LIU*, Microsoft Research Asia
JUNHAN YANG†, University of Science and Technology of China
ZHICHENG DOU, Gaoling School of Artificial Intelligence, Renmin University of China
XING XIE, Microsoft Research Asia
JI-RONG WEN, Beijing Key Laboratory of Big Data Management and Analysis Methods, Key Laboratory of Data Engineering and Knowledge Engineering, MOE

Deep semantic matching aims to discriminate the relationship between documents based on deep neural networks. In recent years, it becomes increasingly popular to organize documents with a graph structure, then leverage both the intrinsic document features and the extrinsic neighbor features to derive discrimination. Most of the existing works mainly care about how to utilize the presented neighbors, whereas limited effort is made to filter appropriate neighbors. We argue that the neighbor features could be highly noisy and partially useful. Thus, a lack of effective neighbor selection will not only incur a great deal of unnecessary computation cost, but also restrict the matching accuracy severely.

In this work, we propose a novel framework, **C**ascaded **D**eep **S**emantic **M**atching (**CDSM**), for accurate and efficient semantic matching on textual graphs. CDSM is highlighted for its two-stage workflow. In the first stage, a lightweight CNN-based ad-hod neighbor selector is deployed to filter useful neighbors for the matching task with a small computation cost. We design both one-step and multi-step selection methods. In the second stage, a high-capacity graph-based matching network is employed to compute fine-grained relevance scores based on the well-selected neighbors. It is worth noting that CDSM is a generic framework which accommodates most of the mainstream graph-based semantic matching networks. The major challenge is how the selector can learn to discriminate the neighbors' usefulness which has no explicit labels. To cope with this problem, we design a weak-supervision strategy for optimization, where we train the graph-based matching network at first and then the ad-hoc neighbor selector is learned on top of the annotations from the matching network. We conduct extensive experiments with three large-scale datasets, showing that CDSM notably improves the semantic matching accuracy and efficiency thanks to the selection of high-quality neighbors. The source code is released at https://github.com/jingjyyao/CDSM.

CCS Concepts: • **Information systems** → *Language models*; *Similarity measures*;

Additional Key Words and Phrases: Semantic Matching, Textual Graph, Neighbor Selection

*Corresponding author.
†Work was done during Junhan Yang's internship in Microsoft.

Authors' addresses: Jing Yao, jingyao@microsoft.com, Microsoft Research Asia, Beijing, 100080; Zheng Liu, zhengliu@microsoft.com, Microsoft Research Asia, Beijing, 100080; Junhan Yang, t-junhanyang@microsoft.com, University of Science and Technology of China, Hefei, 230026; Zhicheng Dou, dou@ruc.edu.cn, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, 100872; Xing Xie, xingx@microsoft.com, Microsoft Research Asia, Beijing, 100080; Ji-Rong Wen, jrwen@ruc.edu.cn, Beijing Key Laboratory of Big Data Management and Analysis Methods, Key Laboratory of Data Engineering and Knowledge Engineering, MOE, Beijing, 100872.

2157-6904/2021/8-ART111 $15.00
https://doi.org/10.1145/3366423.3380294

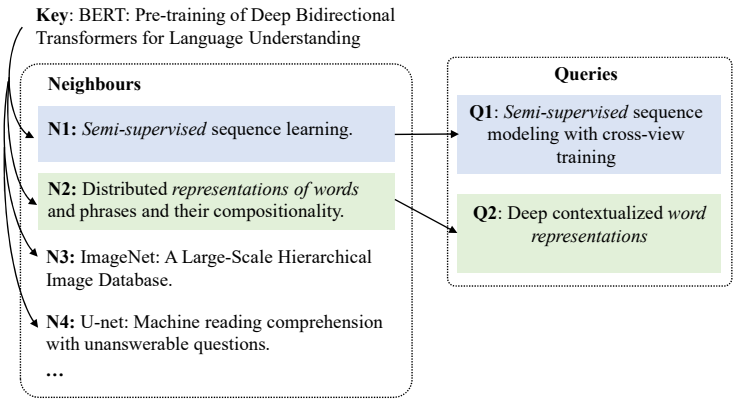

Fig. 1. Example of ad-hoc neighbor selection. The key document is linked to multiple neighbors (papers in its reference) of different semantics. While estimating its relationship with Q1, N1 will be selected as both documents are about *semi-supervised learning*; while estimating the relationship with Q2, N2 will be selected due to the correlation with *word representation*.

ACM Reference Format:
Jing Yao, Zheng Liu, Junhan Yang, Zhicheng Dou, Xing Xie, and Ji-Rong Wen. 2021. CDSM: Cascaded Deep Semantic Matching on Textual Graphs Leveraging Ad-hoc Neighbor Selection. *ACM Trans. Intell. Syst. Technol.* 37, 4, Article 111 (August 2021), 24 pages. https://doi.org/10.1145/3366423.3380294

1 INTRODUCTION

Deep semantic matching aims to discriminate the relationship between documents based on deep neural networks [5, 9, 20]. It plays a critical role in today's intelligent web services, such as recommendation systems, search engines, and online advertising systems [26]. Conventional semantic matching models mainly analyze the intrinsic features of the documents. Thanks to the increasing capacity of deep neural networks, a series of advanced document encoders have been proposed, especially those based on pre-trained language models [11, 15, 17].

1.1 Semantic Matching on Textual Graphs

Many textual datasets can be naturally organized with graph structures, e.g., the webpages of online products can be linked based on users' web browsing behaviors, and the academic literature can be linked based on their citation relationships. In recent years, it becomes increasingly popular to discriminate the matching relationship between two documents on such textual graphs [13, 27, 31, 32], where both the intrinsic document features and the extrinsic neighbor features are jointly leveraged. Existing works mainly emphasize the design of graph-based matching models, whereas limited effort is dedicated to the selection of appropriate neighbors. It's usually assumed that most of the neighbors are informative, and semantic matching may always benefit from the incorporation of neighbor features. As a result, existing methods [27, 32] either use all the available neighbors if the computation capacity allows, or heuristically sample a subset of neighbors for efficient computation.

We argue that **the neighbors could be highly noisy and partially useful, thus appropriate neighbor selection is vital for graph-based semantic matching**. Contradicted to the common assumption, we empirically find the following properties. *1) The neighbor features can be highly **noisy**: in many situations, only a small fraction of neighbors could actually contribute to the current semantic matching. 2) The neighbor feature's usefulness is **conditional**: a neighbor may contribute to*

the semantic matching given one particular target document, but it may become useless when dealing with another target document (as Example 1.1). As such, a lack of effective neighbor selection will severely restrict the performance of graph-based semantic matching. The irrelevant neighbors will not only take a huge amount of unnecessary computation cost, but also introduce strong background noise which deteriorates the matching accuracy.

1.2 Our Work

In this paper, we propose the **C**ascaded **D**eep **S**emantic **M**atching (**CDSM**) framework to address the above challenges. It completes the semantic matching task in two consecutive steps. First, an ad-hoc neighbor selection step is performed to select the optimal subset of neighbors w.r.t. the given documents. By focusing on these selected neighbors which are truly informative, the subsequent semantic matching step can be accomplished with both high accuracy and high efficiency, as background noise and unnecessary computation costs from the irrelevant neighbors can be avoided.

• **Ad-hoc Neighbor Selection**. In CDSM, the neighbor selection is performed in an ad-hoc manner. It is empirically found that merely a small number of neighbors may accurately contribute to the semantic matching task between two specific documents; and an informative neighbor in one matching task will probably become useless in another task[1]. Therefore, static neighbor selection will be inappropriate. Instead of assigning fixed neighbors to each document, CDSM makes **Ad-hoc** neighbor selection, i.e., neighbors are selected w.r.t the counterpart to be matched. Specifically, given a document and its matching counterpart, CDSM aims to identify the neighbors that are closely related to the counterpart. Only the neighbors relevant to the counterpart will be preserved for the subsequent semantic matching task, while the irrelevant ones will be filtered out. We use a concrete example to illustrate the underlying intuition.

Example 1.1. *As shown in Figure 1, the key document "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" should be linked to all documents within its reference list. N1 is about "semi-supervised learning", and N2 is about the "representation of words"; meanwhile, there are many more neighbors about other topics, like "ImageNet dataset", "machine reading comprehension", etc. To determine whether the query document Q1 has a citation link with the key document, the ad-hoc neighbor selector will choose N1 to contribute useful information, rather than N2. Both N1 and Q1 are about "semi-supervised learning". Based on such supporting evidence, the connection between the key document and Q1 can be positively determined with high confidence. However, given another query document Q2, the neighbor N2 becomes a plausible choice. Both Q2 and N2 are about "word representation", based on which the connection between the key document and Q2 can be positively determined with high confidence.*

The neighbor selector is also designed to be **lightweight**. Given that each document may be linked to a vast number of neighbors, it will be infeasible to identify their utilities with heavy-loaded functions. In CDSM, the neighbor selector makes use of lightweight estimation networks, whose inference cost will be small. It is also experimentally validated that the highly simplified backbone networks are already sufficiently accurate to identify useful neighbors. As a result, the computation overhead of neighbor selection will be small enough, making CDSM comparably efficient as the existing methods based on heuristic neighbor sampling.

The subsequent semantic matching is performed based on the well-selected neighbors from the first stage. As discussed, both accuracy and efficiency of the semantic matching will benefit from such a selection, thanks to the elimination of background noise and unnecessary computation costs. It is worth noting that CDSM is a **Generic** framework, where the mainstream graph-based

[1] check Section 3 for the empirical analysis

matching models (e.g., the recent works which combine GNNs and pre-trained language models [13, 31, 32]) can be seamlessly incorporated as the backbone of the matching network.

• **Training via weak-supervision**. One major challenge for CDSM is that there are no explicit measurements of the neighbors' usefulness, which hinders the training of the neighbor selector. In our work, we develop a weak-supervision strategy, where the neighbor selector is trained on top of weak annotations obtained from the semantic matching model. Intuitively, given a pair of documents Q and K, a useful neighbor of Q should provide additional evidence to support the correlation between Q and K. In other words, a useful neighbor will strengthen the correlation between Q and K. Based on such an intuition, we make a formalized definition, which serves as the criterion of whether a neighbor is useful in a specific semantic matching task.

Definition 1.1. Given a pair of documents Q and K, and the function $\mathcal{M}(\cdot)$ which measures the semantic correlation between Q and K. A neighbor $Q.N_i$ is useful, if the semantic correlation between Q and K is improved when Q is aggregated with $Q.N_i$: $\mathcal{M}(Q \oplus Q.N_i, K) > \mathcal{M}(Q, K)$ ("$\oplus$" is the aggregation operator).

With the definition of neighbor usefulness, we develop a weak-supervision algorithm for CDSM. First, the graph-based semantic matching network $\mathcal{M}$ is trained with the neighbors sampled by heuristics. Second, the well-trained semantic matching network makes annotation for the usefulness of each neighbor $Q.N_i/K.N_i$. Finally, the neighbor selector is trained based on the annotation results, where it learns to discriminate the highly useful neighbors from the less useful ones.

Extensive experimental studies are conducted with three large-scale datasets: DBLP, Wiki and Bing Ads[2]. The CDSM's effectiveness is verified from two perspectives. Firstly, as a generic framework, it notably improves the accuracy of a variety of graph-based semantic matching models. Secondly, it achieves competitive running efficiency as high-quality neighbors can be effectively selected with a small computation cost.

To summarize, our major contributions are listed as follows:

- We identify the importance of neighbor selection in the graph-based semantic matching task.
- We propose a novel generic framework CDSM. With the selection of high-quality neighbors, CDSM achieves both high accuracy and high efficiency for graph-based semantic matching.
- We design the weak-supervision strategy to effectively train the neighbor selector on top of the semantic matching network's annotations.
- We perform comprehensive experimental studies, whose results verify the effectiveness and efficiency of CDSM as a generic framework.

The rest of the paper is organized as follows. The related works are reviewed in Section 2. The graph-based semantic matching problem is formally defined and empirically analyzed in Section 3. The detailed methodology of CDSM is elaborated in Section 4. The experimental studies are discussed in Section 5 and 6. Finally, the whole work is concluded in Section 7.

2 RELATED WORKS

Semantic matching between documents is a fundamental problem in information retrieval and recommendation systems. Conventional works mainly rely on the intrinsic document features, e.g., bag-of-words [18], latent semantic analysis [12] and topic modeling [3]. In recent years, various neural text encoders have been extensively explored for this task. In [20, 28], convolution neural networks are utilized to capture the local contextual patterns of the input texts; and in [16, 19], recurrent neural networks are employed to encode the sequential relationship between tokens.

[2]The first two are widely used open benchmark datasets, and the third one is a massive industrial dataset collected by Bing Search.

The latest works are usually built upon the large-scale pre-trained language models, like BERT and RoBERTa [4, 14, 17]. The semantic matching accuracy can be significantly improved thanks to the high-quality deep contextualized text representations.

Actually, many textual datasets can be organized in the form of a graph. For example, in sponsored search, advertisements can be linked together based on users' co-click behaviors [13, 32]. Similarly, online articles and webpages can be connected as a graph according to their mutual linkage relationships. In recent years, it becomes increasingly popular to conduct fine-grained semantic matching on such textual graphs [1, 8, 13, 23, 27, 31, 32], leveraging both the intrinsic document features and the extrinsic neighbor features. The typical approaches [6, 30] will firstly encode each document node into latent representations; and then make use of graph neural networks to aggregate the graph neighbor information. The latest works usually combine graph neural networks with pre-trained language models [13, 32], where the underlying semantics of each individual document can be captured more effectively. To facilitate the in-depth interaction between the textual features and graph structures, Yang et al. [27] propose GraphFormers: the GNN components are nested into each layer of the transformer. In such a way, each document's representation can be contextualized by involving the graph information.

Despite the achieved progress so far, these works mainly focus on designing a better model to aggregate existing neighbors (all neighbors or those selected heuristically), whereas limited effort is made for the selection of truly informative neighbors. Empirically, we have found that 1) the neighbor features can be noisy and 2) the usefulness of a neighbor is conditional, different for different matching counterparts. In this paper, we are committed to solving the problem of effective neighbor selection to improve both the accuracy and efficiency of graph-based semantic matching.

3 PROBLEM DEFINITION

This section covers the following issues: 1) the definition of graph-based semantic matching, 2) the empirical analysis of how neighbor selection matters in graph-based semantic matching, and 3) the definition of neighbor selection, which plays a central role in our CDSM framework.

3.1 Graph-based Semantic Matching

Semantic matching is a critical issue in information retrieval and natural language processing. Given a query document Q and a key document K, a semantic matching model predicts the relevance score between the two documents based on their textual features. Since many real-world data can be organized as graphs, the graph-based semantic matching goes beyond by leveraging the textual features from both the target nodes (i.e. Q and K) and their linked neighbors on the graph. In this place, we define the graph-based semantic matching task in the form of a typical ranking problem.

Definition 3.1. (Graph-based Semantic Matching) Given a pair of documents: query Q and key K, together with their neighbor sets $Q.N$ and $K.N$, the graph-based semantic matching model $\mathcal{M}$ learns to predict the relevance score between the query and key as:

$$m(Q, K) = \mathcal{M}(Q \oplus Q.N, K \oplus K.N), \tag{1}$$

("$\oplus$" is the aggregation operator), such that a positive key K^+ can be ranked higher than a negative key K^-: $m(Q, K^+) > m(Q, K^-)$.

Although the neighbor features may provide complementary information to the semantic matching task, they should be utilized with caution due to the following defects. First, the neighbor features are prone to strong noise: many of the neighbors may contribute little useful information to the semantic matching task. Secondly, the usefulness of a neighbor is conditional: a neighbor can be helpful to the semantic matching task between a query and a specific key document, but

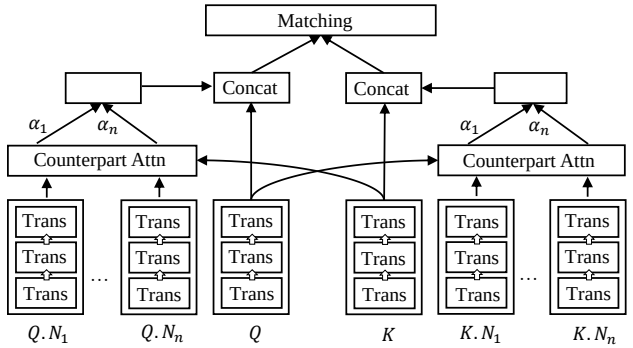

Fig. 2. TextGNN with counterpart attention. The center document Q, K and their neighbors are encoded by the pre-trained language model (with stacked transformers) at first; the neighbors are then aggregated based on the attention of the matching counterpart.

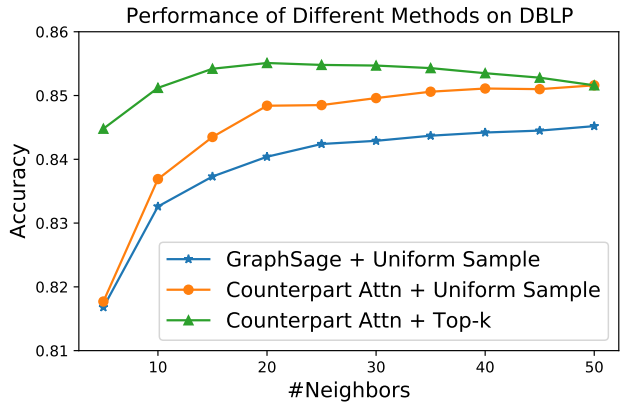

Fig. 3. Case analysis on DBLP. The horizontal axis shows the number of neighbors utilized in each semantic matching task, and the vertical axis shows the semantic matching accuracy (measured by precision@1).

turns useless when dealing with other keys. In the following discussion, an empirical analysis is presented to demonstrate the neighbors' impacts.

3.2 Neighbors' Impact: An Empirical Analysis with TextGNN

We take one of the latest graph-based semantic matching methods, TextGNN [32], for our analysis. The TextGNN framework uses the pre-trained language model BERT as the text encoder, and graph neural networks (like GraphSage) as the graph aggregator. The input documents, i.e. the query/key and their neighbors, are encoded by the text encoder at first. Then, the text embeddings are aggregated by the graph aggregator to get the final representation for semantic matching. In our work, we make an analysis based on the following adaptations of TextGNN.

First, we introduce the "counterpart attention" as shown in Figure 2. The "counterpart" means the document to be matched: the query's counterpart is the key, while the key's counterpart is the query. To identify the neighbors that truly contribute to the current matching task, the neighbors

are scored and aggregated based on their attention with the counterpart document:

$$\text{AGG}(N) = \sum_i \alpha_i * \theta_{N_i},\ \alpha_i = \text{CP} - \text{ATT}(\theta_{N_i}, \theta_{CP}). \tag{2}$$

$\text{AGG}(\cdot)$ is the neighbor aggregation function and CP-ATT$(\cdot)$ means the counterpart attention. θ_{N_i} and θ_{CP} are the embeddings of the neighbor N_i (could be $Q.N_i$ or $K.N_i$) and the counterpart document respectively. With this adaptation, TextGNN becomes more robust to noisy neighbors, as truly useful neighbors for the counterpart can be highlighted with higher attention weights.

Second, we select a subset of neighbors for graph-based semantic matching. A neighbor is selected if it is ranked within the "Top-k" positions w.r.t. the counterpart attention scores. By this means, the quality of neighbor features can be improved as the potentially irrelevant neighbors can be largely removed, therefore leading to higher semantic matching accuracy.

We perform experimental analysis with the DBLP dataset (refer to Section 5 for details). A total of three alternatives are compared: (1) The original TextGNN with GraphSage aggregator (using mean-pooling for implementation), and uniformly sampled neighbors. (2) The adapted TextGNN with counterpart attention, and uniformly sampled neighbors. (3) The adapted TextGNN with counterpart attention, and the Top-k neighbors. All neighbors are sampled from the same candidate set, and each document owns at most 50 neighbors. The analysis results are shown in Figure 3. The semantic matching accuracy (measured by Precision@1, with 1 positive key and 29 negative keys) is checked when different numbers of neighbors are incorporated. The following properties can be observed from the shown results.

• There is indeed strong noise within the neighbor features. The adapted TextGNN with Top-k neighbors consistently outperforms the one with uniformly sampled neighbors, which indicates that the subset of Top-k neighbors is more useful. Besides, the accuracy of TextGNN with the Top-k neighbors reaches its apex when k is merely around 20; it goes down when more neighbors are introduced. It is probably because the additional neighbors (thereafter the apex) are less relevant, which provides little useful information but noisy features to the semantic matching task.

• The usefulness of neighbors is conditional, dependent on the given counterpart. For both methods with the uniformly sampled neighbors, the adapted TextGNN with counterpart attention outperforms the original TextGNN consistently. Such an observation indicates that a neighbor is likely to contribute more to the semantic matching task if it is relevant to the matching counterpart.

Both findings are consistent with our statements about the neighbor features: noisy and conditionally useful. Without an effective neighbor selection mechanism, the graph-based semantic matching could be inaccurate and inefficient due to the introduction of useless neighbors. Notice that although the "counterpart attention based Top-k neighbor selection" improves the accuracy, it has low feasibility in practice as the entire neighbors still need to be encoded by heavy-loaded text encoders. A practical selection mechanism will be introduced in our subsequent discussion.

3.3 Neighbor Selection

Given the discussed properties of the neighbor features, it is important to select the subset of neighbors which are truly helpful to the specific semantic matching task. In this paper, we aim to learn a neighbor selector to complete the selection defined as follows.

Definition 3.2. (Neighbor Selection) Given the semantic matching task between the query document Q and key document K (whether K is positive or negative is unknown), the neighbor selector $\mathcal{S}(\cdot)$ learns to identify the subsets of neighbors:

$$Q.\tilde{N} \leftarrow \mathcal{S}(Q.N|Q,K),\ K.\tilde{N} \leftarrow \mathcal{S}(K.N|Q,K); \tag{3}$$

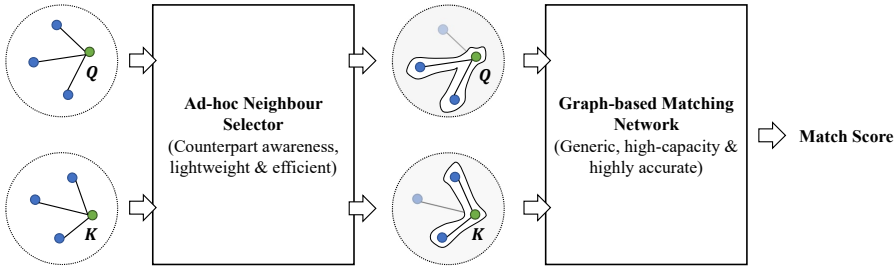

Fig. 4. The whole architecture of our CDSM framework with a two-stage workflow: (1) A lightweight ad-hoc neighbor selector is deployed to filter the optimal subset of neighbors for the matching documents; (2) A high-capacity matching model is employed to calculate the relevance score with these selected neighbors.

such that the semantic matching accuracy can be optimized on top of the selection result: $Q \oplus Q.\tilde{N}$ and $K \oplus K.\tilde{N}$.

According to the above definition, the selector $\mathcal{S}(\cdot)$ is expected to realize the following two functionalities. (1) **Ranking**: which identifies the relative importance of the neighbors; (2) **Truncation**: which determines how many top-ranked neighbors should be selected and utilized. Insufficient selection will be less informative, while excessive selection will introduce noise.

Besides, the selector is desirable for satisfying two properties. **Ad-hoc**: instead of assigning a fixed set of neighbors to each document, the selection needs to be dynamically dependent on the given counterpart: the query's neighbors are selected w.r.t. the key to be matched, and vice versa. **Lightweight**: the neighbor selector needs to be highly efficient, such that it could traverse all neighbors for the optimal subset with affordable computation overhead.

4 CASCADED DEEP SEMANTIC MATCHING

In this work, we propose Cascaded Deep Semantic Matching (CDSM), a generic framework for accurate and efficient semantic matching on textual graphs. It consists of two modules: the lightweight ad-hoc neighbor selector, and the high-capacity graph-based semantic matching network. Besides, we design a hybrid optimization algorithm to train the above two modules. We also demonstrate a prototype implementation of CDSM, which realizes the properties required in Section 3 and shows strong empirical performance.

4.1 The CDSM Framework

The proposed CDSM framework is illustrated in Figure 4, which completes the graph-based semantic matching in two consecutive steps. First, a lightweight ad-hoc neighbor selector is developed to filter the optimal subset of useful neighbors that would help to discriminate the semantic relationship between the query and key. Second, a high-capacity graph-based matching network is employed to calculate the relevance score based on the given documents and their selected neighbors. The detailed formulations are introduced as follows.

4.1.1 Ad-hoc Neighbor Selector. In the first stage, the CDSM takes the query document Q, key document K and all their neighbors $Q.N = [Q.N_1, \ldots, Q.N_n]$, $K.N = [K.N_1, \ldots, K.N_n]$ as the input. Then, the ad-hoc neighbor selection is performed with two operations: 1) ranking, which estimates and compares the importance of all the neighbors, and 2) truncation, which determines how many top-ranked neighbors should be selected (i.e, the value of k when making Top-k selection).

• **Ranking function**. We discuss two optional forms of the ranking function. The first one is the *one-step* ranking function $\mathcal{R}^{one}(\cdot)$, where the importance of a query's neighbor $Q.N_i$ and a

key's neighbor $K.N_j$ is computed as follows:

$$\begin{aligned} r(Q.N_i) &= \mathcal{R}^{one}(Q.N_i|Q,K); \\ r(K.N_j) &= \mathcal{R}^{one}(K.N_j|Q,K). \end{aligned} \tag{4}$$

Because the computation of the neighbor's usefulness $r(Q.N_i)$ and $r(K.N_j)$ purely relies on the Q and K, the importance scores of all neighbors $Q.N$ and $K.N$ can be simultaneously computed in one step. The top-k neighbors can be directly selected thereafter:

$$\begin{aligned} Q.\tilde{N} &= \text{top-}k(r(Q.N_i)|\forall Q.N_i \in Q.N); \\ K.\tilde{N} &= \text{top-}k(r(K.N_j)|\forall K.N_j \in K.N). \end{aligned} \tag{5}$$

The other option is the *multi-step* ranking function, which consecutively filters useful neighbors one by one in multiple steps. In this method, the importance score of a neighbor is estimated based on not only the matching documents Q and K, but also the already selected neighbors $Q.\tilde{N}$ and $K.\tilde{N}$ (initialized to be empty at the beginning):

$$\begin{aligned} r(Q.N_i) &= \mathcal{R}^{mul}(Q.N_i|Q,K,Q.\tilde{N}); \\ r(K.N_j) &= \mathcal{R}^{mul}(K.N_j|Q,K,K.\tilde{N}). \end{aligned} \tag{6}$$

The underlying intuition is that the selected neighbors are desired of providing comprehensive information about Q and K's relationship. As a result, their underlying semantics are desired to be diversified. In other words, it is unnecessary to choose multiple identical or highly similar neighbors, because no additional information can be introduced by them. In this place, the selection is performed for k consecutive rounds w.r.t to the already selected neighbors. The neighbor with the largest information gain is selected in each step:

$$\begin{aligned} Q.\tilde{N} \leftarrow Q.\tilde{N} + Q.N_*, Q.N \leftarrow Q.N \setminus Q.N_*, \quad & \text{where } Q.N_* = \text{argmax}(\text{r}(Q.N_i)); \\ K.\tilde{N} \leftarrow K.\tilde{N} + K.N_*, K.N \leftarrow K.N \setminus K.N_*, \quad & \text{where } K.N_* = \text{argmax}(\text{r}(K.N_i)). \end{aligned} \tag{7}$$

• **Truncation**. The ranking function only measures the relative importance of all neighbors, whereas it is still unclear how many top-ranked neighbors should be selected. Some documents may have very few useful neighbors for the given semantic matching task, while others may have a large number of useful neighbors. In this paper, we propose the following heuristic strategies to determine how to truncate the top-ranked neighbors to get the optimal subset, including basically static methods and more flexibly dynamic methods. All these strategies can be easily applied together with the ranking functions. In practice, the truncation strategy can be chosen based on empirical performance.

Fixed Capacity. The simplest way of truncation is to set a fixed capacity and always select the same number of neighbors for all documents when making the Top-k selection. As discussed above, a fixed capacity may lack the flexibility to deal with documents with different numbers of useful neighbors. Thus, more complicated strategies are further introduced for a complement.

Absolute Value as Threshold. Since the utility of neighbors is reflected by the predicted score $r(Q.N_i)$ or $r(K.N_j)$, we think there should be an absolute score threshold τ to distinguish whether a neighbor is necessary to be selected. In this case, a neighbor $Q.N_i$ will be selected if $r(Q.N_i) > \tau$, the same for $K.N_j$. Both approaches set a static threshold (capacity or value) for truncation, more flexibly dynamic methods are listed in the next.

Overall Ranking. Recall that we define the semantic matching task in a typical ranking style as Definition 1.1, we think the truly positive key documents would have more useful neighbors to support their connection with the query document than those negative ones. Thus, when measuring the utility of all keys' neighbors with the ranking function, the positive key counterpart would

have more neighbors that achieve high ranking scores, and more neighbors should be selected for it. Based on this hypothesis, we propose to rank the neighbors of all candidate keys in a unified list, and select those neighbors within the Top-p position. We illustrate the matching tasks between a query document Q and a series of key documents $K_1, K_2, \ldots$ as an example. We copy Q for many times to construct document pairs with all the keys, obtaining $[Q_1, K_1], [Q_2, K_2], \ldots$ where all Q_i are the same. For all keys, we rank all their neighbors $K_i.N_j$ in a unified list based on the ranking score $r(K_i.N_j)$ and selected those neighbors within the Top-p position for the corresponding key. With regard to the copied queries $Q_1, Q_2, \ldots$, we rank all their neighbors $Q_i.N_j$ in an unified list on top of their ranking scores $r(Q_i.N_j)$ and select the Top-p neighbors. Although the Q and $Q.N$ are copied, when dealing with different key documents, the ranking score of the query neighbors is decided by different keys to highlight different neighbors.

Relevance Score as Threshold. Another intuition is that the introduction of neighbor features should strengthen the correlation between Q and K, as presented in Definition 1.1. Therefore, we expect that the selected neighbors can outscore the original relevance between Q and K. At first, we measure the Q and K's similarity as $sim(Q, K)$ (sample implementation of $sim(Q, K)$ will be introduced in Subsection 4.3). Then, a neighbor $Q.N_i/K.N_i$ will be selected if it satisfies $r(Q.N_i) > sim(Q, K)/r(K.N_i) > sim(Q, K)$.

To summarize, the method of fixed capacity selects the same number of neighbors for all documents, while the other three methods are adaptive to different matching tasks. The two kinds of strategies can be combined together to benefit each other.

4.1.2 Graph-based Matching Network. In the second stage, a graph-based semantic matching network is deployed. The matching network will take the query Q, key K, and the selected neighbors $Q.\tilde{N}$ and $K.\tilde{N}$ as the input. Then, it predicts Q and K's relevance score as:

$$m(Q, K) = \mathcal{M}(Q \oplus Q.\tilde{N}, K \oplus K.\tilde{N}), \tag{8}$$

where "$\oplus$" is the graph aggregator. Though CDSM is generic and adaptive to various matching models, we desire a high-capacity graph-based semantic matching network in order to derive fine-grained relevance prediction, e.g., TextGNN and GraphFormers. (This is different from the neighbor selector which requires lightweight computation.) Besides, knowing that the matching network only needs to deal with a small set of well-selected neighbors, $|Q.\tilde{N}| \ll |Q.N|$ and $|K.\tilde{N}| \ll |K.N|$, the relevance prediction is ensured to be made with much less computation overhead.

4.2 The CDSM Optimization

The optimization of CDSM is challenging because there are no explicit measurements of neighbors' usefulness. To deal with this challenge, we propose a hybrid optimization workflow, where the neighbor selector is trained on top of weak annotations from the semantic matching network.

• **Hybrid Optimization** The proposed hybrid optimization works in the following three steps, as illustrated in Figure 5. First, we train the graph-based matching network $\mathcal{M}$ based on the labeled document pairs and randomly sampled neighbors. Second, the well-trained semantic matching model is used to annotate the neighbor's usefulness based on Definition 1.1. Finally, the ad-hoc neighbor selector is trained on the annotation results through contrastive learning. The details of each step are illustrated in the following.

• **Train Semantic Matching Network**. Given a positive pair of documents Q, K and their neighbors $Q.N, K.N$ which are selected by random sampling or heuristic rules, the matching score is calculated as $m(Q, K) = \mathcal{M}(Q \oplus Q.N, K \oplus K.N)$ (Eq. 8). Then, we randomly sample a batch of negative key documents $[K_1^-, K_2^-, \ldots, K_M^-]$. The matching network is trained to distinguish the

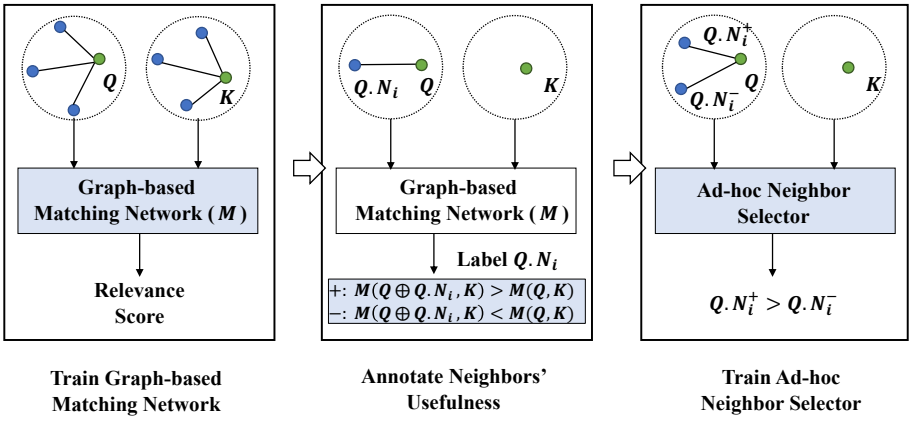

Fig. 5. The hybrid optimization workflow of the CDSM framework: (1) Training the graph-based matching network $\mathcal{M}$ based on the document pairs with heuristically sampled neighbors; (2) Annotating the neighbors' usefulness with the well-trained matching network; (3) Training the neighbor selector with these weak annotations.

positive keys from the negative keys by minimizing the classification loss.

$$\mathcal{L}^{cls} = -\frac{\exp(m(Q,K^+))}{\exp(m(Q,K^+)) + \sum_i \exp(m(Q,K_i^-))}. \tag{9}$$

In our implementation, we make use of "in-batch negative samples" [10, 15] to reduce the computation cost, where for one positive document pair, the keys of the other samples in the same mini-batch are viewed as negative keys.

• **Train Ad-hoc Neighbor Selector**. The neighbor selector is trained via weak supervision, with neighbors' usefulness annotated by the semantic matching model trained in the last step. Based on whether the one-step or multi-step ranking function is utilized, the data annotation is performed in two different ways.

When the one-step ranking function $R^{one}(\cdot)$ is utilized, a neighbor's usefulness is determined purely based on the input features of Q and K. For a positive pair of documents Q, K, their semantic matching score without any neighbor features can be calculated as $\mathcal{M}(Q,K)$. Then, the usefulness of each query neighbor $Q.N_i$ for this matching task is evaluated as $\mathcal{M}(Q \oplus Q.N_i, K)$, and that of each key neighbor $K.N_i$ is denoted as $\mathcal{M}(Q, K \oplus K.N_i)$. A neighbor is useful if it improves the matching network's prediction about the relationship between Q and K, based on which the labels are generated by the formulations:

$$\begin{aligned} l(Q.N_i) &= \begin{cases} +: \ \mathcal{M}(Q \oplus Q.N_i, K) > \mathcal{M}(Q,K), \\ -: \ \mathcal{M}(Q \oplus Q.N_i, K) \leq \mathcal{M}(Q,K), \end{cases} \\ l(K.N_i) &= \begin{cases} +: \ \mathcal{M}(Q, K \oplus K.N_i) > \mathcal{M}(Q,K), \\ -: \ \mathcal{M}(Q, K \oplus K.N_i) \leq \mathcal{M}(Q,K). \end{cases} \end{aligned} \tag{10}$$

$l(Q.N_i)$ and $l(K.N_i)$ are the labels of the neighbors, and $\mathcal{M}(\cdot)$ denotes the matching network's predicting function.

When the multi-step ranking function $R^{mul}(\cdot)$ is utilized, the neighbor's usefulness is measured based on not only Q and K, but also the already selected neighbors. As a result, we define a neighbor to be useful if it further improves the matching network's prediction about Q and K's relevance

after introducing it into the currently selected neighbor subset $Q.\tilde{N}, K.\tilde{N}$.

$$
\begin{aligned}
l(Q.N_i) &= \begin{cases} +: \ \mathcal{M}(Q \oplus \{Q.N_i, Q.\tilde{N}\}, K) > \mathcal{M}(Q \oplus Q.\tilde{N}, K), \\ -: \ \mathcal{M}(Q \oplus \{Q.N_i, Q.\tilde{N}\}, K) \leq \mathcal{M}(Q \oplus Q.\tilde{N}, K), \end{cases} \\
l(K.N_i) &= \begin{cases} +: \ \mathcal{M}(Q, K \oplus \{K.N_i, K.\tilde{N}\}) > \mathcal{M}(Q, K \oplus K.\tilde{N}), \\ -: \ \mathcal{M}(Q, K \oplus \{K.N_i, K.\tilde{N}\}) \leq \mathcal{M}(Q, K \oplus K.\tilde{N}). \end{cases}
\end{aligned} \tag{11}
$$

With the neighbors of all positive document pairs annotated, we construct neighbor pairs comprised of a positive neighbor N_i^+ and a negative neighbor N_i^- to train the selector by contrastive learning. Taking a given query document Q as an example, the neighbor selector is learned by minimizing the following loss:

$$
\mathcal{L}^{sel} = -\frac{1}{1 + \exp(-(r(Q.N_i^+) - r(Q.N_i^-)))}. \tag{12}
$$

$r(Q.N_i^+)$ and $r(Q.N_j^-)$ are calculated by the corresponding $R^{one}(\cdot)$ or $R^{mul}(\cdot)$, when different ranking functions are applied.

4.3 The CDSM Implementation

In this place, we show a prototype implementation of the ad-hoc neighbor selector, which will be used in our experiments. In order to guarantee efficiency, we adopt a lightweight architecture, including a CNN-based text encoder and a ranking function. The implementation of graph-based semantic matching networks follows baseline models, whose details will be introduced in Section 5.

As for the CNN-based text encoder, the input is an individual text, taking the query $Q = [w_1^Q, w_2^Q, \ldots]$ as an example. The first layer is a word embedding layer that converts tokens into low-dimensional vectors. By passing the query through this layer, we obtain a word embedding matrix $V^Q = [v_1^Q, v_2^Q, \ldots]$. The second layer is a 1-d CNN layer, capturing local context information within the text sequence to obtain better word representations. As for the i-th term, its context-aware representation c_i^Q is calculated as:

$$
c_i^Q = \text{ReLU}(F_w \times v_{(i-k):(i+k)}^Q + b_w), \tag{13}
$$

where $v_{(i-k):(i+k)}^Q$ means the word embeddings from the position $(i-k)$ to $(i+k)$. $2k+1$ is the size of the context window. F_w and b_w are the parameters of CNN filters. Through the 1-d CNN layer, the output is a matrix of contextual word representations, denoted as $C^Q = [c_1^Q, c_2^Q, \ldots]$. Considering that different words in a sentence contribute different informativeness, we set the third layer as a word-level attention network. The query in the attention mechanism is a trainable dense vector q_w. We compute the attention weight of each word based on the interaction between the query q_w and the context-aware word representation, i.e.,

$$
\alpha_i = \frac{\exp(\alpha_i)}{\sum_{j=1}^{|Q|} \exp(\alpha_j)}, \quad \alpha_i = (c_i^Q)^T \tanh(W_q \times q_w + b_q). \tag{14}
$$

Finally, the text representation r^Q of Q is the weighted sum of all word representations based on the attention weights, as:

$$
r^Q = \sum_{i=1}^{|Q|} \alpha_i c_i^Q. \tag{15}
$$

Using this CNN-based text encoder, we are able to obtain the text representation for the matching documents Q, K and all the presented neighbors.

Table 1. Time complexity analysis of CDSM and other graph-based matching methods.

Method	All Neighbors	Heuristic Selection	CDSM
Time Complexity	$n * T_m$	$n * T_h + k * T_m$	$n * T_s + k * T_m$

With Q, K and all neighbors represented as vectors, the ranking function is employed on them to evaluate the neighbors' importance. With regard to the one-step ranking function, we calculate the usefulness of a neighbor as the relevance between it and the matching counterpart:

$$\mathcal{R}^{one}(Q.N_i|Q,K) = (r^{Q.N_i})^T r^K, \tag{16}$$

$$\mathcal{R}^{one}(K.N_i|Q,K) = (r^{K.N_i})^T r^Q. \tag{17}$$

As for the multi-step ranking function, it calculates the usefulness for a neighbor $Q.N_i/K.N_i$ according to both the selected neighbor set $Q.\tilde{N}/K.\tilde{N}$ and the matching documents Q, K. To highlight the different aspects of information that the already selected neighbors cover, we perform max-pooling along the last dimension to aggregate all selected neighbors. Thus, the usefulness is calculated as:

$$\mathcal{R}^{mul}(Q.N_i|Q,K,Q.\tilde{N}) = \varphi([r^Q, \text{max-pool}(r^{N_i}|\forall N_i \in N)])^T r^K, \tag{18}$$

where $N = Q.\tilde{N} + Q.N_i$. $\varphi(\cdot)$ is an MLP layer, $[\cdot,\cdot]$ indicates vector concatenation, and max-pool$(\cdot)$ is the max-pooling operation along the last dimension.

For the function $sim(\cdot)$ to measure Q and K's similarity, it works as:

$$sim(Q,K) = (r^Q)^T r^K. \tag{19}$$

4.4 The CDSM Efficiency Analysis

As stated in Section 1, filtering a set of irrelevant neighbors with a lightweight neighbor selector could improve both the efficiency and accuracy of graph-based semantic matching. Here, we deploy a mathematical time complexity analysis to prove the CDSM efficiency. We compare the computation time for the semantic matching task of directly using all neighbors, sampling neighbors heuristically and selecting neighbors with our CDSM framework. The comparison results are displayed in Table 1.

In Table 1, T_m represents the time cost of a high-capacity semantic matching network to encode a document. T_s indicates the time cost of the lightweight selector to process a neighbor node ($T_s << T_m$). T_h means the time cost of selecting a neighbor through heuristic methods, such as random sampling. n and k represent the number of all available neighbors and that of the selected neighbors respectively. As such, when $k << n$, CDSM saves a lot of time for computing the fine-grained document representations compared with using all neighbors. Our designed lightweight selector obtains document representations through CNN, thus the time cost T_s would be so small that CDSM can be comparably efficient as the heuristic sampling methods.

In the next sections, we conduct experiments to verify the strong performance and efficiency of the above introduced simplified selector.

5 EXPERIMENTAL SETTINGS

5.1 Datasets and Evaluations

5.1.1 Datasets. We use three large-scale textual graph datasets for evaluation. 1) **Wikidata5M**[3], a million-scale knowledge graph dataset introduced in [24]. It contains entities and their connections. Each entity has an aligned passage from the corresponding Wikipedia page and we take the first

[3]https://deepgraphlearning.github.io/project/wikidata5m

Table 2. Statistics of the three datasets.

Dataset	**Wikidata5M**	**DBLP**	**Product**
#Item	4,018,299	3,691,796	2,049,487
Avg.#N	27.35	46.94	17.02
#Train	7,145,834	3,009,506	3,004,199
#Valid	66,167	60,000	50,000
#Test	100,000	100,000	536,575

sentence from the passage as its textual description. 2) **DBLP**[4]. This dataset contains academic papers up to 2020-04-09 collected from DBLP[5]. The connections between different papers are built upon their citation relationships. We use each paper's title as its textual description. 3) **Product**. This is a product graph dataset constructed from the real-world Bing search engine. We track each user's web browsing events along with the targeted product web pages and divide her continuous events into sessions with 30 minutes of user inactivity as interval [29]. Following a typical product graph construction method in e-commerce platforms [22], the products within a common session are linked to each other. Each product has its corresponding textual description which indicates the product name, brand, type and so on.

For these three datasets, we uniformly sample at most 50 neighbors for each center node from the set comprised of its one-order and two-order neighbors. The statistics of datasets are listed in Table 2.

5.1.2 Evaluations. In this paper, we define the graph-based semantic matching task in the form of a ranking problem, as Definition 3.1. For each positive pair of query and key document, we sample 29 negative key documents, and the target is to rank the positive key higher than these negative keys. Two common ranking metrics are leveraged to evaluate the matching accuracy: **precision@1 (p@1)** and **ndcg**.

5.2 Baselines

Our proposed CDSM is a generic framework to improve both the accuracy and efficiency of the graph-based semantic matching task. It is highlighted for the two-stage workflow instead of a specific semantic matching model: a lightweight neighbor selector to obtain informative neighbors and then the graph-based semantic matching task is conducted, where various mainstream graph-based matching models can be adapted.

5.2.1 Graph-based Semantic Matching Models. To verify its generality, various mainstream graph-based semantic matching models are incorporated as the backbone of the matching network. In these models, texts are all encoded with BERT. The last layer's [CLS] token embedding is treated as the document representation.

• **BERT** [4]: It calculates the semantic matching score as the dot product between the query's and key's [CLS] vectors encoded by BERT respectively, without any neighbor features.

• **TextGNN** [32]: This is one of the most recent graph-based semantic matching methods, with pre-trained language model BERT as the text encoder and GNNs to aggregate the information from neighbors. We consider the following forms of GNNs. **GAT** [21], which combines neighbors and the center document as a weighted sum of all their text vectors. The weight of each text vector is calculated as the attention score with the center document. **GraphSage** [7], where the neighbors are

[4]https://originalstatic.aminer.cn/misc/dblp.v12.7z
[5]https://dblp.uni-trier.de/

first aggregated, then the aggregation result is concatenated with the center document embedding and passed through a dense layer to generate the final representation for semantic matching. There are four different aggregation variants: **MaxSage** and **MeanSage**, which aggregate neighbors by max-pooling and mean-pooling respectively. **AttnSage** aggregates the neighbors based on their attention scores with the center document. **Counterpart** aggregates the neighbors based on their attention scores with the matching counterpart, as presented in Section 3. The above variants of TextGNN are denoted as **TextGNN(GAT/Max/Mean/Attn/CP)** in our experiments.

• **BERT4Graph**: This is a naive extension of BERT from traditional text matching to graph-based semantic matching. For the query, it concatenates the tokens of the query and all neighbors, with a [CLS] token added to the head. Then, the joint token sequence is inputted into BERT for fine-grained interactions. The last layer's [CLS] token embedding is used for the semantic matching, the same for the key document. The semantic matching score is calculated as their cosine similarity.

• **GraphFormers** [27]: This model adopts a deeper fusion of GNNs and text encoders than TextGNN to make more effective use of neighbor features. It nests GNN components between the transformer layers to conduct iterative text encoding and neighbor aggregation.

Except that BERT uses no neighbor information, the other matching models consider neighbors. We apply them as the backbone of the matching network, and compare their performance under different neighbor selection methods.

5.2.2 Selection Approaches. Furthermore, to verify the effectiveness of the ad-hoc neighbor selector under our CDSM framework, we compare it with other selection approaches, including random sampling and two heuristic rule-based methods. All these baselines assign a static set of neighbors to each document, without dynamically considering the matching counterparts like our CDSM. Details are listed as follows.

• **Random Sampling**: For each document, a set of distinct neighbors is randomly sampled from all the presented neighbors.

• **Popularity**: For each document, we rank all its neighbors according to their popularity, i.e., how many nodes they have connections with. Then, the top-k popular neighbors are selected.

• **Similarity**: For each center document, the top-k neighbors that are most similar to it are selected. The similarity is computed as the dot product between their vectors encoded by BERT.

• **CDSM**: This indicates the ad-hoc neighbor selector under our framework.

5.3 Model Settings

In our experiments, UniLMv2-base [2] is applied as the backbone of BERT. As for the ad-hoc neighbor selector, the word embedding layer is initialized with the underlying word vectors in UniLMv2-base. Dimensions of the word embeddings and hidden states are set as 768. We utilize the uncased WordPiece [25] to tokenize all text sequences. The max length of sequences is set as 64 for Wikidata5M, 32 for DBLP, 32 for Product. For the CNN text encoder, the size of the context window is 3. The batch size is 300 and the learning rate is 1e-5. Adam optimizer is applied for training the selector.

As for high-capacity matching models, on Wikidata5M, DBLP and Product, the batch size is 160, 240, 240; the learning rates are 5e-6, 1e-6, 1e-5. 'In-batch negative samples' and Adam are used for optimization. Each training sample is comprised of two groups of documents: 1 query document with 5 randomly sampled neighbors; and 1 key document with 5 randomly sampled neighbors. The training is conducted with 8x Nvidia V100-16GB GPUs.

6 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we conduct extensive experiments to explore the following research questions:

Table 3. Overall performance of different graph-based semantic matching models with different available neighbor sets: all neighbors, randomly sampled 5 neighbors, 5 neighbors selected according to the popularity, the similarity with the center document and the usefulness measured by CDSM. "†" denotes that the result is significantly better than other selection methods (except for all neighbors) in t-test with $p < 0.05$ level. "*" denotes that the result is significantly better than that with all neighbors in t-test with $p < 0.05$ level. Results of the best neighbor selection method are shown in bold.

Data	Methods	All		Random (5)		Popularity (5)		Similarity (5)		CDSM (5)	
		p@1	ndcg	p@1	ndcg	p@1	ndcg	p@1	ndcg	p@1	ndcg
Wiki	BERT	.644	.817	-	-	-	-	-	-	-	-
	TextGNN(GAT)	.580	.772	.521	.732	.416	.664	**.587**	**.781**	.581	.778
	TextGNN(Mean)	.694	.850	.678	.840	.627	.810	.662	.828	**.694**†	**.850**†
	TextGNN(Max)	.687	.844	.676	.839	.653	.827	.666	.831	**.697**†	**.852**†
	TextGNN(Attn)	.696	.850	.679	.840	.620	.803	.664	.829	**.694**†	**.850**
	TextGNN(CP)	.705	.856	.681	.841	.679	.843	.674	.836	**.704**†	**.855**†
	BERT4Graph	-	-	.690	.846	.622	.789	.683	.841	**.705**†	**.858**†
	GraphFormers	.704	.854	.689	.845	.698	.852	.685	.841	**.707**	**.860**
DBLP	BERT	.821	.914	-	-	-	-	-	-	-	-
	TextGNN(GAT)	.826	.918	.776	.892	.790	.900	.851	.929	**.860**†*	**.936***
	TextGNN(Mean)	.845	.908	.817	.894	.816	.894	.815	.891	**.844**†	**.907**†
	TextGNN(Max)	.805	.888	.812	.891	.813	.892	.812	.890	**.833**†*	**.902**†*
	TextGNN(Attn)	.843	.906	.815	.892	.813	.892	.813	.875	**.842**†	**.906**†
	TextGNN(CP)	.852	.917	.819	.901	.818	.901	.816	.899	**.850**†	**.916**†
	BERT4Graph	-	-	.864	.936	.873	.941	.878	.942	**.901**†	**.955**†
	GraphFormers	.903	.955	.877	.944	.882	.946	.880	.942	**.909**†	**.959**†
Product	BERT	.481	.707	-	-	-	-	-	-	-	-
	TextGNN(GAT)	.568	.774	.578	.781	.545	.760	.567	.770	**.588***	**.787**
	TextGNN(Mean)	.746	.883	.723	.868	.721	.867	.703	.852	**.748**†	**.885**†
	TextGNN(Max)	.730	.873	.728	.871	.721	.867	.712	.858	**.756**†*	**.890**†*
	TextGNN(Attn)	.744	.882	.725	.869	.722	.867	.706	.847	**.741**†	**.881**
	TextGNN(CP)	.756	.889	.728	.871	.723	.867	.710	.857	**.760**†	**.892**†
	BERT4Graph	-	-	.739	.877	.733	.873	.722	.864	**.771**†	**.898**†
	GraphFormers	.737	.879	.715	.864	.711	.862	.701	.852	**.759**†*	**.891**†

- **RQ1**: As a generic framework, can CDSM consistently improve both the accuracy and efficiency of the semantic matching task when combined with various graph-based semantic matching networks? (Discussed in Subsection 6.1)
- **RQ2**: Different truncation approaches are proposed to determine the number of selected neighbors. How do they impact the effect and efficiency of the semantic matching task respectively? (Discussed in Subsection 6.2)
- **RQ3**: Can the simplified selector learn to identify the usefulness of neighbors precisely enough? (Discussed in Subsection 6.3)
- **RQ4**: How about the effect and efficiency of the one-step and multi-step ranking functions in CDSM? (Discussed in Subsection 6.4)

6.1 Overall Performance (RQ1)

To demonstrate the generality and effectiveness of CDSM, we compare the performance of different semantic matching models in various scenarios where different neighbor sets are available: all

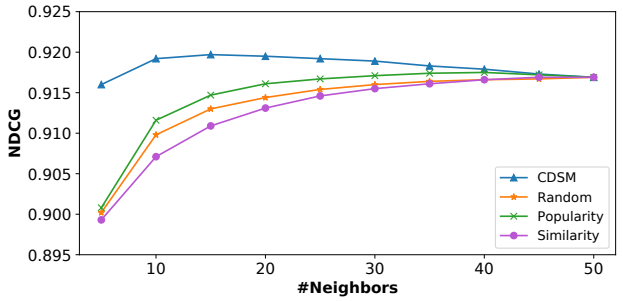

Fig. 6. Performance curves of the TextGNN(CP) model with different numbers of neighbors selected by different methods.

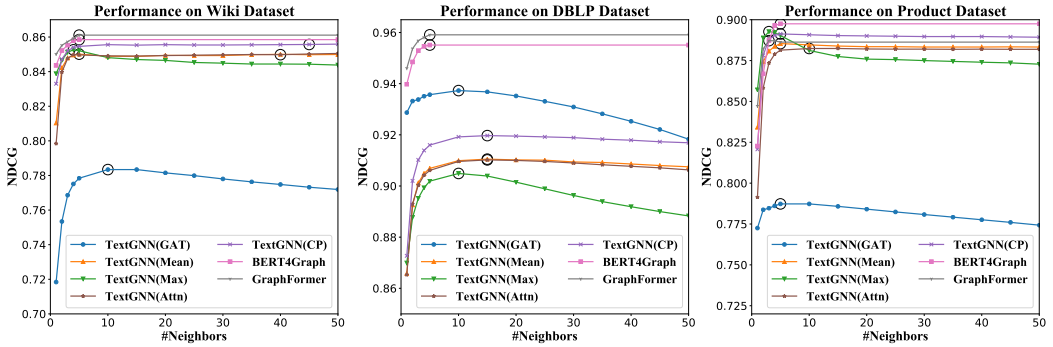

Fig. 7. Performance curves of different semantic matching models with different number of neighbors selected by CDSM. The best results are marked by black circles.

neighbors, 5 randomly sampled neighbors, 5 most popular neighbors, 5 neighbors most similar with the center document and 5 neighbors filtered by the ad-hoc neighbor selector in CDSM. The comparison results are presented in Table 3. We also conduct sensitivity testing with TextGNN(CP) on the number of neighbors selected by different methods, shown in Figure 6. Furthermore, we check the matching accuracy with the top k neighbors selected by CDSM as k becomes larger, shown in Figure 7. Due to the limitation of computing resources, at most 5 neighbors are tested for BERT4Graph and GraphFormers. Observing all these results, we have several findings:

First, **extrinsic neighbor features provide helpful information for the semantic matching task, but these neighbor features are indeed highly noisy. Selecting an optimal neighbor subset can achieve better performance than using all neighbors.** With joint consideration of the center document and neighbors, these graph-based semantic matching models outperform the BERT only baseline in most cases. BERT4Graph and GraphFormers which make fuller use of the neighbor features achieve the best results. However, only a small fraction of neighbors actually contribute to the semantic matching accuracy while others are noise. Comparing the results of different graph-based matching models with all neighbors and the best-performed 5 neighbors (shown in bold) on all datasets in Table 3, most matching models present similar performance. This result demonstrates that a lot of neighbors take no information to improve the matching accuracy. Focusing on the performance curves displayed in Figure 7, the curve of most models rises first and then goes down as more and more neighbors with little informativeness are

involved. This trend proves that selecting a set of truly informative neighbors can achieve better matching accuracy than utilizing all neighbors.

Second, **our proposed lightweight ad-hoc neighbor selector has the ability to efficiently select the truly informative neighbors to enhance the semantic matching accuracy.** As shown in Table 3, the 5 neighbors selected by our CDSM framework perform much better than the neighbor subsets constructed by other selection baselines. Random sampling is unable to distinguish the usefulness of neighbors. In Popularity and Similarity, a static set of neighbors with a specific attribute is assigned to each document and used in matching tasks with various counterparts, thus leading to negative performance. Especially for the Popularity method, a popular neighbor node on the textual graph may be general and contain less informativeness to help specific document matching tasks, which is obvious on the Wiki dataset. Differently, our neighbor selection is performed in an ad-hoc manner to filter neighbors that are relevant to the current counterpart to facilitate the current matching task. Figure 6 further presents the comparison between different selection methods with different numbers of neighbors. CDSM could always filter more informative neighbors to achieve better results than other methods, especially for cases where the computation resources are limited and the number of available neighbors is small. From Figure 7, we can come to that the neighbors ranked at high positions by CDSM significantly improve the semantic matching accuracy, whereas the neighbors with lower scores hardly improve the accuracy and even reduce the results. This observation proves that our selector can precisely measure the usefulness of neighbors for a specific matching task. Besides, we also test the time and space cost of neighbor selections. Table 5 shows that the neighbor selector is sufficiently lightweight and efficient.

Third, **our generic CDSM framework can be well combined with various graph-based matching models to consistently achieve better effects and efficiency on the semantic matching task.** As shown in Table 3, for different graph-based matching models, our CDSM framework can consistently select an effective neighbor subset to achieve comparable or better results than using all neighbors. This verifies the generality of our proposed CDSM framework. In Figure 7, we find that the best results of different models are all achieved with a small subset of informative neighbors selected by our selector, instead of encoding all the presented neighbors. Thus, a large amount of unnecessary encoding cost is saved and much background noise is avoided to get better results. Observing Table 5, little selection cost is increased while large encoding cost can be reduced by selecting a subset of neighbors by CDSM. Therefore, both the effects and efficiency of semantic matching can be improved with the CDSM framework.

To conclude, we confirm that **neighbor selection is critical for the graph-based semantic matching task. Our CDSM framework can efficiently select a more effective neighbor subset for the matching task, improving both effectiveness and efficiency.**

6.2 Analysis of Truncation (RQ2)

As for the ad-hoc neighbor selector in our CDSM framework, in addition to the ranking function that predicts the relative importance of all neighbors, we set truncation to determine how many top-ranked neighbors should be selected. Several different truncation approaches are proposed. We take empirical experiments to analyze their impacts on the effects and efficiency of semantic matching respectively. The results are displayed in Table 4. For all cases, the max number of selected neighbors is restricted to 20.

First of all, setting a relatively small fixed capacity as the truncation shows very stable performance on various matching models. For the documents with few informative neighbors, these neighbors are involved without too much noise. For those documents with a lot of effective neighbors, selecting a part of top-tanked neighbors may be sufficient to distinguish them from the

Table 4. Comparison of truncation methods. "Avg.N" indicates the average number of selected neighbors.

Datasets	Methods	Fixed Capacity		Absolute Score		Overall Ranking		Relevance Score	
		NDCG	Avg.N	NDCG	Avg.N	NDCG	Avg.N	NDCG	Avg.N
Wiki	TextGNN(GAT)	.784	14.00	.784	7.09	**.785**	**4.04**	.784	5.79
	TextGNN(Mean)	.850	6.00	.850	4.23	**.851**	**5.62**	.850	9.99
	TextGNN(Max)	.853	5.00	.852	2.99	.852	2.80	**.853**	**3.22**
	TextGNN(Attn)	.850	7.00	.850	5.71	.850	5.71	**.851**	**10.28**
	TextGNN(CP)	.856	20.00	.855	10.33	.856	14.85	**.856**	**10.66**
DBLP	TextGNN(GAT)	.937	10.00	**.938**	**10.13**	.938	11.06	.937	6.80
	TextGNN(Mean)	.911	16.00	.910	11.06	.911	11.80	**.911**	**8.13**
	TextGNN(Max)	.905	11.00	.901	8.95	.905	8.47	**.905**	**7.32**
	TextGNN(Attn)	.910	15.00	.910	12.93	.910	12.47	**.910**	**8.87**
	TextGNN(CP)	.920	18.00	.919	14.28	.920	14.32	**.920**	**9.48**
Product	TextGNN(GAT)	.788	7.00	**.788**	**2.27**	.788	3.36	.788	4.77
	TextGNN(Mean)	.885	6.00	.885	4.71	**.886**	**3.27**	.886	3.91
	TextGNN(Max)	.893	3.00	.893	2.56	.892	1.76	**.893**	**2.55**
	TextGNN(Attn)	.883	8.00	.883	5.82	**.883**	**4.05**	.882	4.46
	TextGNN(CP)	.892	6.00	.892	4.67	**.892**	**3.31**	.891	4.34

Table 5. Comparison about the time and space cost of the lightweight selector and high-capacity semantic matching network.

Scenarios	All 50 Neighbors		Select 5 Neighbors	
	Time(ms)	Space(MB)	Time(ms)	Space(MB)
Selection	-	-	1.72	1000
Matching	1425.0	10219	217.0	2361
Total	1425.0	10219	218.72	3361

negative keys. Introducing adaptive strategies further reduces the computation cost, even promoting the matching accuracy. By setting an appropriate absolute value threshold, we could adaptively select informative neighbors for each pair of matching documents. Fewer neighbors will be selected for the scenarios where only a small fraction of neighbors are informative, instead of a fixed number, thus saving a lot of computing resources. More neighbors would be selected for the documents that have many informative neighbors, providing more supporting evidence to enhance the matching. The relevance score threshold shows the best performance in most cases. Recall that in Definition 1.1, we argue that a neighbor contributes to a specific document matching task when it provides additional evidence. As such, the relevance score threshold performs as a serious filter to highlight neighbors that are more informative than the center document.

6.3 Effects and Efficiency of Selector (RQ3)

To ensure the running efficiency, the neighbor selector in our CDSM framework should be lightweight. We compare its computation cost with the high-capacity matching network and present the results in Table 5. The time and space cost of the selector are both much smaller than the expensive matching model, which could be ignored. Then, we conduct an experiment to analyze whether the highly simplified selector can learn to accurately identify useful neighbors.

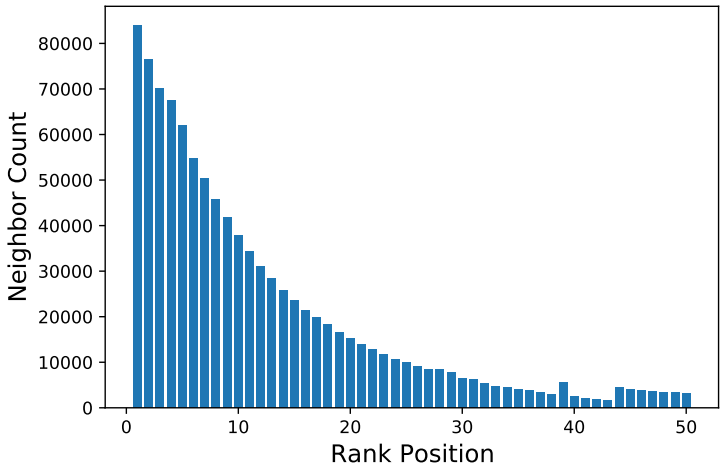

Fig. 8. Distribution of the top 10 neighbors selected in the CDSM framework. The rank position is decided by the usefulness annotated by the well-trained semantic matching model.

Due to a lack of labels to explicitly measure the usefulness of neighbors, we develop a weak-supervision strategy to train the neighbor selector on top of weak annotations generated by the well-trained matching model. Thus, we evaluate the learning ability of the simplified selector by how well it could adapt to the annotations from the matching model. We make a comparative analysis of their respective evaluations about the neighbors. At first, we rank all neighbors of each matching task according to their usefulness measured by the semantic matching model as Definition 1.1. Then, we apply the trained selector to score the neighbors and observe which neighbors in the above ranking are scored as the top 10. The distribution of original ranking positions is displayed in Figure 8. We find that most of the top 10 neighbors scored by the selector still correspond to those ranked at the top 10 positions by the high-capacity matching model, some at 10-20. This demonstrates that the highly simplified selector fits well to the annotations provided by the high-capacity matching model. It can identify useful neighbors efficiently and accurately.

6.4 One-Step v.s. Multi-Step Selection (RQ4)

As for the ad-hoc neighbor selector in CDSM, we propose two different ranking functions, i.e. one-step and multi-step ranking functions. The one-step ranking function measures the usefulness of all neighbors based on only Q, K and selects the top-k highest-scoring neighbors at one step. The multi-step ranking function ranks and selects neighbors one by one. In each step, the neighbor leading to the largest information gain is selected. We experiment to compare these two selectors with TextGNN. Limited by the computing resources, at most 5 neighbors are selected.

Observing the performance curve of the two ranking functions in Figure 9, both approaches have the ability to distinguish the informativeness of different neighbors and select effective neighbors for the matching task. The multi-step ranking function performs a little better than the one-step. We infer that it may be because the multi-step method selects neighbors in a greedy way to construct a best-performed neighbor subset, instead of evaluating each neighbor separately. In this case, the computation cost of multi-step selection would be much larger than the one-step selection since it needs to evaluate the neighbors multiple times. On the whole, the more efficient one-step selection can achieve a better trade-off between effects and efficiency. We employ the one-step ranking function in other studies.

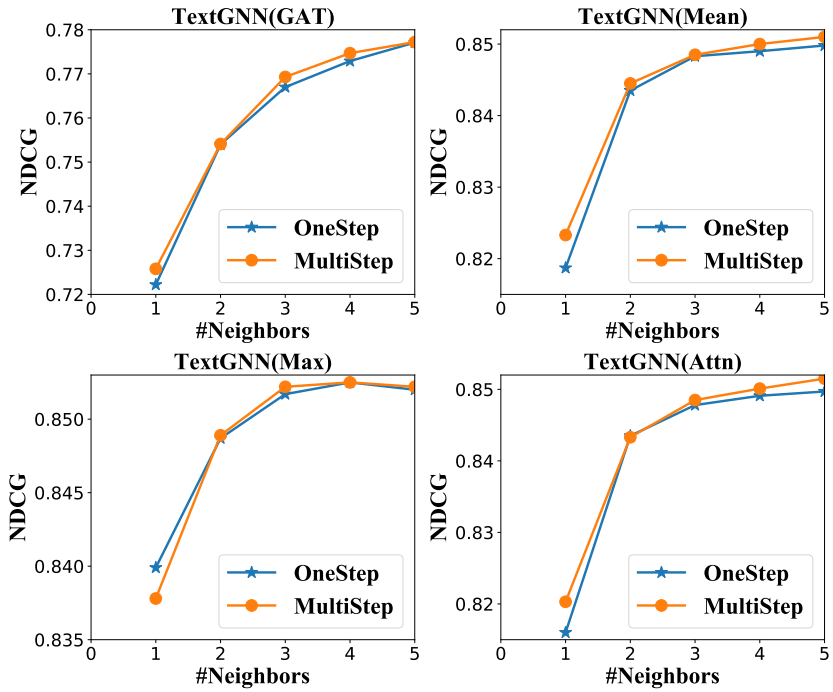

Fig. 9. Performance curves of One-Step Selector and Multi-Step Selector on the Wiki dataset.

6.5 Case Study

The neighbor selector is the core module in our proposed CDSM framework, and the quality of selected neighbors would directly affect the accuracy of the subsequent text matching. In order to evaluate the quality of neighbor selection, we carry out a qualitative case study on the DBLP dataset. For each pair of documents (Query Q, Key K) to be matched, we adopt the neighbor selector to score all neighbor nodes. Then, we compare the information contributed to the current matching task by the top 5 neighbors and the worst 5 neighbors respectively. Experimental results are presented in Figure 10.

We make the following observations. As for the neighbors highly scored by the selector, they can contribute some valuable information for determining the relationship between the current query and key documents. For example, facing the query "Reinforcement learning: a survey", the neighbor selector highlights the node relevant with "reinforcement learning" for discrimination. As for the query "'DeepLab: Semantic Image Segmentation...', neighbors related to image segmentation are selected. However, the low-scoring neighbor nodes are much less relevant to the query but only relevant to the key, and even introduce noisy signals.

7 CONCLUSION

In this paper, we identify the importance of neighbor selection in the graph-based semantic matching task and propose a novel framework, Cascaded Deep Semantic Matching (CDSM). It leverages a two-stage cascaded workflow for the semantic matching task. A lightweight neighbor selector is employed to efficiently filter informative neighbors for the given matching documents in the first stage. Then, the matching model calculates fine-grained relevance scores based on these selected neighbors. Both higher matching accuracy and speed are achieved by our CDSM framework,

Query	Key	Selected Key Neighbors	Discarded Key Neighbors
Reinforcement learning: a survey	Autonomous development of vergence control driven by disparity energy neuron populations	1. Fully automatic extraction of salient objects from videos in near real-time	1. Automatic object segmentation with salient color model
		2. Decentralized Supervisory Control of Discrete Event Systems Based on Reinforcement Learning	2. Fast Video Classification via Adaptive Cascading of Deep Models
		3. Olfactory search at high Reynolds number	3. Boundedly Rational Agents Playing the Social Actors Game: How to Reach Cooperation
		4. Predicting Future Instance Segmentations by Forecasting Convolutional Features	4. A Teleoperation System Utilizing Saliency-Based Visual Attention
		5. An Experimental Study of Adaptive Control for Evolutionary Algorithms	5. The role of orientation diversity in binocular vergence control
DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs	R-CASENet: A Multi-category Edge Detection Network	1. Selective Removal of Impulse Noise Preserving Edge Information	1. Deep multi-center learning for face alignment
		2. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network	2. DeepEdge: A multi-scale bifurcated deep network for top-down contour detection
		3. Context-Aware Synthesis for Video Frame Interpolation	3. Edge Focusing
		4. GraspFusion: Realizing Complex Motion by Learning and Fusing Grasp Modalities with Instance Segmentation	4. Feature space learning model
		5. Semi-supervised Segmentation of Optic Cup in Retinal Fundus Images Using Variational Autoencoder	5. R-CASENet: A Multi-category Edge Detection Network

Fig. 10. Case study of the selected and discarded neighbor nodes of the key document for the corresponding query. “yellow” highlights relevance with the query and “blue” highlights relevance with the key.

as unnecessary computation and background noise from irrelevant neighbors are avoided. We develop CDSM as a generic framework that can be seamlessly incorporated with mainstream graph-based models. A weak-supervision strategy is proposed to train the selector with weak annotations generated by the matching model. Empirical experiments on three large-scale textual graph datasets confirm the effectiveness and generality of our CDSM framework.

In summary, one limitation of the current CDSM is that the neighbor selector can accurately compare the usefulness of neighbors to highlight informative ones, but presents weak performance on truncation, i.e. determining how many neighbors should be selected. In Section 4.1, we discuss several truncation approaches, without one consistently performing the best on all datasets. Designing a more universal automatic truncation method is a research topic in the future. In addition, the current two-step optimization method of CDSM can be iterated multiple times. How to enhance the ability of both the selector and matching model through iterative optimization is also under-explored.

REFERENCES

[1] Siddhant Arora. 2020. A Survey on Graph Neural Networks for Knowledge Graph Completion. *CoRR* abs/2007.12374 (2020).

[2] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 642–652.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3 (2003), 993–1022.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186.

[5] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 55–64.

[6] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[7] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 1024–1034.

[8] Linmei Hu, Siyong Xu, Chen Li, Cheng Yang, Chuan Shi, Nan Duan, Xing Xie, and Ming Zhou. 2020. Graph Neural News Recommendation with Unsupervised Preference Disentanglement. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 4255–4264.

[9] 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*. 2333–2338.

[10] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. 6769–6781.

[11] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 39–48.

[12] Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25, 2-3 (1998), 259–284.

[13] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. AdsGNN: Behavior-Graph Augmented Relevance Modeling in Sponsored Search. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 223–232.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019).

[15] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, Dense, and Attentional Representations for Text Retrieval. *Trans. Assoc. Comput. Linguistics* 9 (2021), 329–345.

[16] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24, 4 (2016), 694–707.

[17] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 3980–3990.

[18] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[19] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional Attention Flow for Machine Comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[20] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*. 373–374.

[21] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[22] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 839–848.

[23] Meihong Wang, Linling Qiu, and Xiaoli Wang. 2021. A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry* 13, 3 (2021), 485.

[24] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Trans. Assoc. Comput. Linguistics* 9 (2021), 176–194. https://transacl.org/ojs/index.php/tacl/article/view/2447

[25] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.08144 (2016).

[26] Jun Xu, Xiangnan He, and Hang Li. 2018. Deep learning for matching in search and recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1365–1368.

[27] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, S Amit, Guangzhong Sun, and Xing Xie. 2021. GraphFormers: GNN-nested Transformers for Representation Learning on Textual Graph. *Thirty-Fifth Conference on Neural Information Processing Systems* (2021).

[28] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 1480–1489.

[29] Jing Yao, Zhicheng Dou, and Ji-Rong Wen. 2020. Employing Personal Word Embeddings for Personalized Search. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1359–1368.

[30] 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*. 974–983.

[31] Yusi Zhang, Chuanjie Liu, Angen Luo, Hui Xue, Xuan Shan, Yuxiang Luo, Yiqian Xia, Yuanchi Yan, and Haidong Wang. 2021. MIRA: Leveraging Multi-Intention Co-click Information in Web-scale Document Retrieval using Deep Neural Networks. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 227–238.

[32] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. TextGNN: Improving Text Encoder via Graph Neural Network in Sponsored Search. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 2848–2857.