Syntactic-Informed Graph Networks for Sentence Matching

CHEN XU and JUN XU, Gaoling School of Artificial Intelligence, Renmin University of China, China
ZHENHUA DONG, Noah's Ark Lab, Huawei, China
JI-RONG WEN, Gaoling School of Artificial Intelligence, Renmin University of China, China

Matching two natural language sentences is a fundamental problem in both natural language processing and information retrieval. Preliminary studies have shown that the syntactic structures help improve the matching accuracy, and different syntactic structures in natural language are complementary to sentence semantic understanding. Ideally, a matching model would leverage all syntactic information. Existing models, however, are only able to combine limited (usually one) types of syntactic information due to the complex and heterogeneous nature of the syntactic information. To deal with the problem, we propose a novel matching model, which formulates sentence matching as a representation learning task on a syntactic-informed heterogeneous graph. The model, referred to as SIGN (Syntactic-Informed Graph Network), first constructs a heterogeneous matching graph based on the multiple syntactic structures of two input sentences. Then the graph attention network algorithm is applied to the matching graph to learn the high-level representations of the nodes. With the help of the graph learning framework, the multiple syntactic structures, as well as the word semantics, can be represented and interacted in the matching graph and therefore collectively enhance the matching accuracy. We conducted comprehensive experiments on three public datasets. The results demonstrate that SIGN outperforms the state of the art and also can discriminate the sentences in an interpretable way.

CCS Concepts: • **Information systems** → **Information retrieval**; • **Computing methodologies** → **Natural language processing;**

Additional Key Words and Phrases: Sentence matching, syntactic structures, graph learning

ACM Reference format:
Chen Xu, Jun Xu, Zhenhua Dong, and Ji-Rong Wen. 2023. Syntactic-Informed Graph Networks for Sentence Matching. *ACM Trans. Inf. Syst.* 42, 2, Article 38 (September 2023), 29 pages.
https://doi.org/10.1145/3609795

The work of J. Xu was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education.
This work was funded by the National Key R&D Program of China (2019YFE0198200), Beijing Outstanding Young Scientist Program (BJJWZYJH012019100020098), Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, fund for building world-class universities (disciplines) of Renmin University of China, as well as the Outstanding Innovative Talents Cultivation Funded Programs 2023 of Renmin University of China.
Authors' addresses: C. Xu, J. Xu (corresponding author), and J.-R. Wen, Gaoling School of Artificial Intelligence, Renmin University of China, No. 59 Zhongguancun Street, Haidian District, Beijing, 100872, China; emails: {xc_chen, junxu}@ruc.edu.cn, jrwen@ruc.edu.com; Z. Dong, Huawei Technologies Co Ltd, Bantian Huawei Base, Longgang District, Shenzhen, 518000; email: dongzhenhua@huawei.com.

1046-8188/2023/09-ART38 $15.00
https://doi.org/10.1145/3609795

1 INTRODUCTION

Natural language sentence matching is a funda\mental technique in information retrieval and **Natural Language Processing (NLP)**. Typical tasks include relevance ranking, **Paraphrase Identification (PI)**, and **Natural Language Inference (NLI)**. For example, the matching models for PI have been developed to determine whether two sentences are semantically identical. In NLI, text matching has been used to determine whether a hypothesis is entailment, contradiction, or neutral given a premise.

Many research efforts have been devoted to developing matching models that can bridge the semantic gaps between two sentences [24, 54]. Previously, the matching was conducted based on superficial matching signals such as shared words/phrases, as well as word/phrase embedding similarities and word orders. Typical approaches include the representation-based methods of DSSM [19], C-DSSM [14, 39], the interaction-based methods of ARC-II [18], MatchPyramid [35], DRMM [17], KNRM [52], and their combinations: Duet [32], RE2 [55], and BERT4Match [11]. Recently, studies have shown that NLP knowledge, especially the syntactic structures, is useful in sentence matching. Among them, the syntactic-based representation methods encode the syntax of two sentences *separately* and aggregate them to make the prediction [8, 9, 25, 34]; the syntactic-based interaction methods directly utilize the different syntactic tags as fine-grained matching signals and utilize these signals to conduct sentence matching [6]. Recently, Liu et al. [25], Xu et al. [53], and Zhang et al. [59] proposed to combine the sentence syntax with **Pre-Trained Language Models (PLMs)** and achieved state-of-the-art sentence matching performance.

Although performance improvements have been achieved, one obvious weakness of existing approaches is that they usually use limited (usually one) types of syntactic structures and encode them *separately* for matching. In this way, the rich complementary information contained in other types of syntactic structures is inevitably overlooked. Previous studies of sentence linguistics [13, 16] have verified that all types of syntactic structures are crucial for discovering sentence semantics. In sentence matching, different syntactic structures provide complementary matching signals, which are important for sentence matching. More detailed examples can be found in Section 3.2. However, due to the complex and heterogeneous nature of the syntactic information, it is difficult for existing matching models to utilize the multiple syntactic structures and capture their complementary matching signals.

Given a natural language sentence, several syntactic structures can be parsed in the NLP pipeline process, including the relation structures (e.g., syntactic dependencies, word-chunk structures) and the attribute structures (e.g., **Parts of Speech (POS)**, chunk types, and **Named Entities (NEs)**). Different structures are represented differently. The relation structures represent that one word/chunk syntactic relates to another word/chunk. For example, syntactic dependencies are usually represented as the directed links that connect the words that have one-to-one correspondence. The chunks, however, are usually represented as links from the words to their higher-order units (chunks). The attribute structure reflects the attribute of the word/chunk. For example, POS/NE tags (chunk types) are simply the category labels of the words (chunk). In matching, relation structures provide an approximation to the semantic relationship between prediction and their arguments [8, 34]. Attributes of words/chunks are useful for encapsulating the rich syntax structural patterns [6].

Ideally, to achieve high matching accuracy, a matching model would utilize the syntactic structures and ensure their good interaction. In real practices, however, the complexity and heterogeneity of the syntactic structures hinder their full potential.

In this article, we aim to develop a novel sentence matching model that can leverage multiple types of syntactic structures and make them interact in a fine-grained way. Inspired by the graph learning framework [45, 47], we propose SIGN (Syntactic-Informed Graph Network) to develop a graph learning based model for sentence matching. Specifically, given a pair of sentences, the syntactic structures (e.g., POS and **Named Entity Recognition (NER)** tags, syntactic dependencies, chunking structures) are first extracted with an NLP parser (e.g., using Stanford CoreNLP [31]). Then a heterogeneous matching graph is constructed in which each node corresponds to a word, a chunk, or a sentence. The edges are inferred according to different types of extracted syntactic structures: as for the relation structures, the edge represents that one node syntactic relates to another node; as for attribute structures, and the edge represents one node that has the same attributes as another node. In a graph learning framework, the nodes with edges will interact and their representation will be similar. Therefore, all syntactic structures can be well incorporated into the matching graph and provide complementary matching signals.

In the SIGN training phase, after initializing PLM (e.g., BERT [11] or RoBERTa [28]) embeddings, the **Graph Attention Network (GAT)** [45] or **Heterogeneous Graph Attention Network (HGAT)** [47] is utilized to learn the representations of nodes from different types of edges, followed by a relation classifier for making the final matching prediction. In this way, the syntactic structures, as well as the semantics of the two sentences, are respectively encoded in the edges and nodes of the matching graph and collectively contribute to the final matching.

Incorporating the semantic and syntactic information in one matching model offers several advantages: exploiting rich syntactic information in matching, collectively utilizing the syntactic and semantic information, ease in interpretation, and improved prediction accuracy. The contributions of this work can be summarized as follows:

- We highlight the necessity of simultaneously using semantic information and syntactical information (including the related structures and the attribute structures) in sentence matching. A novel matching model called *SIGN* is proposed in which the matching graph based on syntactic structures enables the model to conduct the text matching in an accurate, robust, and interpretable way.
- Experimental results based on three publicly available benchmarks show that SIGN outperforms the state-of-the-art baselines.
- Analysis shows that SIGN not only can bridge the semantic gaps between sentences but also can discriminate the sentences that are similar in form but different in meaning.

2 RELATED WORK

2.1 Syntactic-Free Sentence Matching Models

Machine learning has been widely adapted to the task of matching natural language sentences [24]. Existing approaches can be categorized as representation-based models, interaction-based models, and their combinations [54]. Representation-based models try to represent the two input sentences in a high-level semantic space and then conduct matching in the space. The early work of DSSM [19] makes use of a deep feed-forward network to respectively represent the inputted sentences as dense vectors (see also [14, 18, 39, 49]). The attention mechanism [36] has also been adopted by matching models to enable the matching model to get a richer representation. For example, the recently developed models of DRCN [43], CSRAN [21], and RE2 [55] all stack the encoding layers and attention layers to obtain the sentence representations for matching. Interaction-based models directly capture the local interaction relationship between the elements (e.g., words, phrases) of two sentences. The interaction can be represented as, for example, the matching matrices or the attention vectors. The final matching scores can be achieved by integrating the local interactions. Representative models in this category include ARC-II [18], MatchPyramid [35], and DeepMatch [29], among others. For example, MatchPyramid [35] uses a CNN to aggregate the local word interaction signals in the matching matrix. DeepMatch [29] is designed for integrating the local interaction signals based on the hierarchical structures of the topics. Researchers also found that combining both representation-based and interaction-based models can achieve better matching accuracy, as reported in DUET [32]. Recently, the PLM of BERT has also been fine-tuned for the matching task, and state-of-the-art performances have been achieved [11, 28]. These traditional models always utilize superficial matching signals (e.g., word-word correspondence and word order) and ignore the implicit or explicit NLP knowledge.

2.2 Syntactic-Based Sentence Matching Models

In recent years, there has been a research trend that utilizes rich NLP knowledge to improve sentence matching, which can be categorized as syntactic-based representation and interaction methods. The syntactic-based representation methods encode the syntax of two sentences *separately* and aggregate them to make the prediction. For example, Liu et al. [27] also proposed an end-to-end neural network to exact the latent constituency trees of sentences and then matched their latent structures by the weighted sum of all possible sentence spans. Potthast et al. [37] employed the n-grams of the syntactic structure sequence and encoded them as matching features for paraphrasing identification. HIM [9] utilized the recurrent neural network (known as the tree-RNN) to aggregate word representation utilizing the relation structures (word syntactic dependencies) (also see [8, 34]). The syntactic-based interaction methods directly utilize the different syntactic tags as fine-grained matching signals, and the tags are often extracted from the attribute structures such as POS, constituent parsing, and NER. For example, Das and Smith [10] and Mohammad et al. [33] utilized the POS tag as syntactic features of classifiers to identify the paraphrase, and MIX [6] utilized the POS tag and NE of two sentences as prior knowledge for multi-layer convolutions. Recently, Sachan et al. [38] found that syntax can help the PLM capture more information and achieve impressive results for NLP tasks. Bai et al. [2] proposed to utilize different semantic role labeling tags and constituency trees to enhance the representation ability of PLMs, respectively. Liu et al. [25] proposed to combine syntactic dependency and semantic role labeling tags to conduct sentence matching and achieved state-of-the-art performances on sentence matching tasks.

2.3 Graph-Based Sentence Matching Models

The graph-based model has also been used for sentence matching. Bu et al. [5] and Wang et al. [46] viewed the sentences as graphs, which can extract and match the subgraph patterns with conventional graph kernels. With the development of the **Graph Convolutional Network (GCN)** [23], GAT [45], and HGAT [47], deep graph learning has been applied to the matching task since it is a more practical way to represent the nodes and edges with embeddings. TextGCN [56] jointly learns the embeddings with the GCN for both various documents and their words through a heterogeneous graph. ED-GAT [30] and DEPGCN [58] view the dependency parse tree as a graph and utilize the GCN to obtain the syntactic embeddings of the input texts. Recently, Sachan et al. [38] proposed a syntax graph based model with a PLM on NLP tasks and proved the effectiveness of syntactic structure on the PLM. Yu et al. [57] utilized the hierarchical matching signals with a GNN model.

3 SYNTACTIC STRUCTURES FOR SENTENCE MATCHING

3.1 Problem Formulation

In matching, given a pair of natural language sentences, their relationship is predicted with a matching function. Formally, suppose that $\mathcal{Z}$ is the set of labels that are defined by a specific matching task. In the preceding PI examples, $\mathcal{Z} = \{0, 1\}$, where 0 and 1 respectively indicate the

Table 1. Notations and Their Explanations

Notations	**Explanations**
$(X, Y, z) \in \mathcal{D}$	sentence pair with label z
$X(\text{or } Y) = \{w_1, w_2, \ldots, w_{t_X}\}$	sentence and corresponding words
$C_X(\text{or } C_Y) = \{c_1, c_2, \ldots, c_{n_X}\}$	chunk sequence of X(or Y)
$\mathcal{G}_{X,Y} = (\mathcal{V}, \mathcal{E}, \mathbf{H})$	matching graph of X and Y
$\mathcal{V} = \mathcal{V}_w \cup \mathcal{V}_c \cup \mathcal{V}_s$	nodes in the matching graph
$\mathcal{V}_w$	word nodes
$\mathcal{V}_c$	chunk nodes
$\mathcal{V}_s$	sentence nodes
$\mathcal{E} = \mathcal{E}_{rel} \cup \mathcal{E}_{att} \cup \mathcal{E}_{agg}$	edges in the matching graph
$\mathcal{E}_{rel}$	relation edges
$\mathcal{E}_{att}$	attribute edges
$\mathcal{E}_{agg}$	aggregation edges
$\{a^i_{v_1}, a^i_{v_2}, \ldots, a^i_{v_t}\}$	i-th attribute tags of node in $\mathcal{V}$
$\mathcal{G}^X_{r_i} = (\mathcal{V}^X_{r_i}, \mathcal{E}^X_{r_i})$	i-th relation structure graph in X
$\mathbf{H} \in \mathbb{R}^{d \times \lvert\mathcal{V}\rvert}$	node representations of $\mathcal{G}_{X,Y}$

labels of "dissimilar" and "similar"; in NLI, $\mathcal{Z} = \{0, 1, 2\}$, where 0, 1, and 2 respectively indicate "contradiction," "neutral," and "entailment." In the training phase, a set of training instances $\mathcal{D} = \{(X_i, Y_i, z_i)\}_{i=1}^N$ is given, where given were each sample $(X, Y, z) \in \mathcal{D}$ consists of a sentence pair (X, Y) as well as its corresponding ground-truth matching label z. Moreover, the sentences X and Y are two sequences of words: $X = \{w^X_1, w^X_2, \ldots, w^X_{t_X}\}$ and $Y = \{w^Y_1, w^Y_2, \ldots, w^Y_{t_Y}\}$, where w^X_i and w^Y_j denote the i-th and j-th words in X and Y, and t_X and t_Y are the number of words (lengths) of X and Y, respectively. The objective is to learn a matching model that takes a sentences pair as input and outputs the prediction of the sentence pair's relationship. Table 1 lists the major notations that will be used in the article.

3.2 Problem Analysis

Existing sentence matching models either consider the inputted sentences pair (X, Y) as sequences of tokens, overlooking the syntactic information, or utilize very limited syntactic information in matching [2, 6, 34, 59]. However, fully exploiting the syntactic structures is important for accurate sentence matching. We will give two examples to show how the multiple different syntactic information helps to identify the semantic dissimilar and semantic similar sentence pairs, which have different syntactic structures.

The first example is shown in Figure 1, which illustrates a sentence pair from the **Quora Question Pairs (QQP)** test set whose ground-truth label $z = 0$ (dissimilar). The parsed syntactic structures, including the relation structure (e.g., syntactic dependency, word-chunk structure) and attribute structure (e.g., chunk type and POS tags), are shown. We can see that the two sentences X and Y are quite similar in terms of the superficial matching signals: they share the most important words w^X_i, w^Y_j: ("what," "is," "the," "program," "meaning," "of," "SAP") after stemming. Moreover, they are similar to the shared bi-grams "what is" and "the meaning of." Additionally, the ordering of words is similar. At the same time, their attribute (e.g., POS, chunk types) syntactic structures are also quite similar. Existing matching models (e.g., [2, 6]) only utilize the attribute knowledge, which results in wrong predictions for the example pair. However, the prediction becomes much easier if the relation structures (e.g., word syntactic dependencies) are taken into consideration: it is easy to know that the first sentence focuses on "workflow" while the second focuses on "SAP."

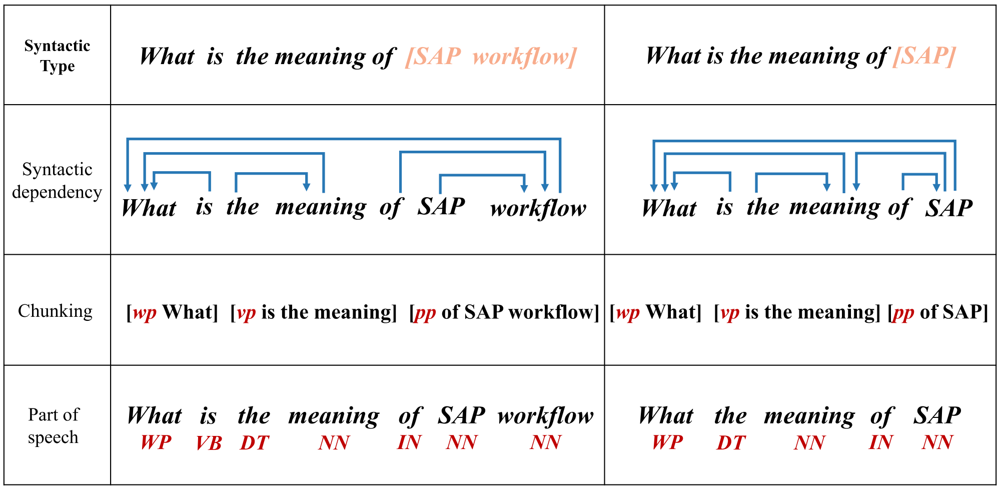

Syntactic Type
What is the meaning of [SAP workflow]
What is the meaning of [SAP]
Syntactic dependency
What is the meaning of SAP workflow
What is the meaning of SAP
Chunking
[wp What] [vp is the meaning] [pp of SAP workflow]
[wp What] [vp is the meaning] [pp of SAP]
Part of speech
What is the meaning of SAP workflow
WP VB DT NN IN NN NN
What the meaning of SAP
WP DT NN IN NN

Fig. 1. Example 1: Syntactic structures for real sentence pairs from the QQP dataset. The ground truth label is "dissimilar."

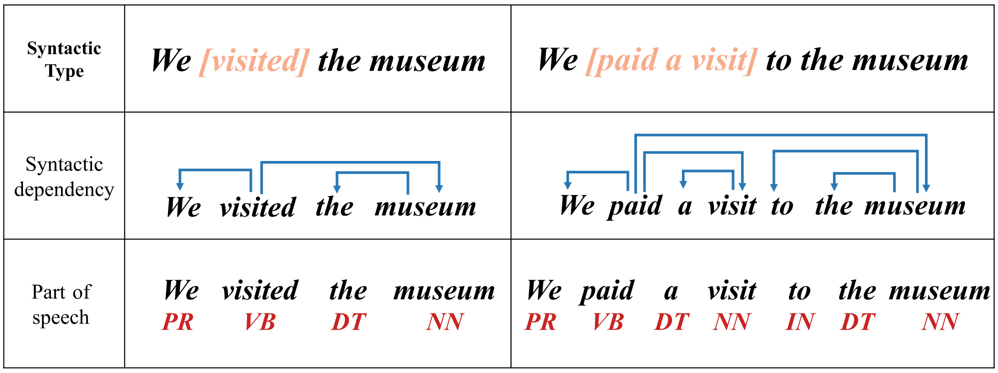


Fig. 2. Example 2: Syntactic structures for real sentence pairs from a general case. The ground truth label is "similar."

The second example shown in Figure 2 illustrates a sentence pair (X, Y) whose ground-truth label $z = 1$ (similar), but they have different syntactic structures. Similarly, the relation structure (syntactic dependency) and attribute structure (POS tags) are shown. We can see that they share similar semantics but have different forms of the words: the word ("visit") in sentence X has the *verb* form, and the word ("visit") in the phrase ("paid a visit") of sentence Y has the *noun* form. However, the sentences share the same semantic meanings, because the word ("visit") in sentence X is the verbalization of the word ("visit") in sentence Y. This verbalization can be categorized as the word form transformation, and it can be easily identified if the POS tags and word syntactic dependencies are taken into consideration. For example, the verbalization of nouns can be identified by the ("VB$\xrightarrow{\text{obj}}$NN" and "NN") structures easily, where "obj" is the word-word dependency type.

Previous studies and recent online paraphrasing tools [12] usually categorized the different syntax transformations into four types: (1) different forms of a word, (2) word/chunk equations or

Table 2. Syntactic Structures for Identifying the Syntax Transformation

Syntax transformation Types	**Explanations**	**Examples**	**Identified Syntactic Structures**
different forms of a word	replacing words using a different form (e.g., verbalization of nouns or adjectivization of verbs)	("We *visited* the museum."; "We *paid a visit* to the museum.")	attribute structures (e.g., POS, chunk tags) and relation structures (e.g., word syntactic dependencies)
word/chunk equations or synonyms	replacing words or chunks with their synonyms	("The cloud is *similar* to the airplane."; "The cloud is *comparative* to the airplane.")	attribute structures (e.g., POS, NE, and chunk tags)
active or passive sentence	adjust the sentences in the active voice to the passive voice, or vice versa	("The dog *is chasing* the cat."; "The cat *is chased* by the dog.")	relation structures (e.g., word syntactic dependencies)
different word/chunk order	preposition or postposition of different components of a sentence	("*At the weekend*, we went hiking."; "We went hiking *at the weekend*.")	attribute structures (e.g., chunk tags) and relation structures (e.g., word syntactic dependencies)

synonyms, (3) active or passive sentence, and (4) different word/chunk order. These syntax transformations aim to change the syntax of a sentence while maintaining the semantics of a sentence. A detailed description and applied syntactic structures can be found in Table 2.

Please note that the preceding two types of examples are not rare in the dataset. Based on the experimental results of BERT (trained on the QQP dataset), we found that in the error cases, the syntactically dissimilar pairs account for 83.9%.

Therefore, we conclude that (1) different sentence syntactic structures are all essential in sentence matching, and (2) it is necessary to fully exploit the syntactic structures because different types of the structures can work collectively and complementarily.

In practice, however, the syntactic structures, including the relation structures and attribute structures, are complex and heterogeneous. Ideally, a matching model would utilize these syntactic structures simultaneously and make them fully interactive.

4 OUR APPROACH: SIGN

In this section, we propose SIGN, a novel syntactic-informed graph network for sentence matching that utilizes multiple syntactic structures in a general way. Next, we first describe the general framework and then explain the model details.

4.1 General Framework

Figure 3 illustrates the proposed SIGN. The left side shows the overall matching process of a sentence pair, and the right side shows the matching graph details. Given a pair of natural language sentences, SIGN first processes the sentence pair with an NLP syntactic parser including the relation structure and attribute structure. After that, the graph representation learning algorithm of the GAT [45] or HGAT [47] is applied over the matching graph, with the embeddings from the PLM as the initial node representations. Finally, the relation classifier is employed to summarize the learned representations and outputs the final matching prediction.

In the following sections, we explain the key components of SIGN in detail.

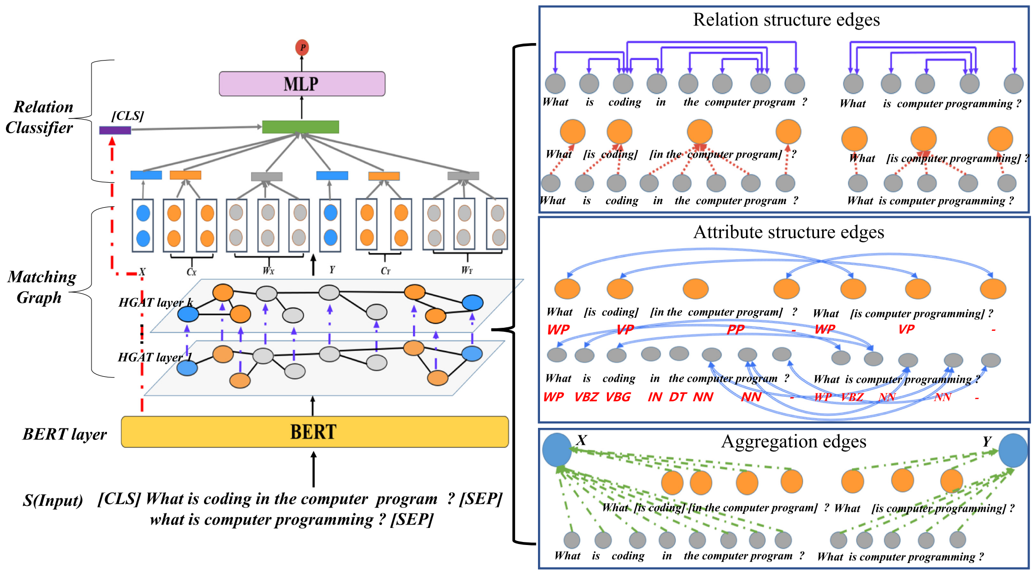

Relation Classifier
Matching Graph
BERT layer
S(Input) [CLS] What is coding in the computer program ? [SEP] what is computer programming ? [SEP]
MLP
[CLS]
HGAT layer k
HGAT layer 1
BERT
Relation structure edges
Attribute structure edges
Aggregation edges
What is coding in the computer program ?
What is computer programming ?
WP VBZ VBG IN DT NN NN
WP VP PP WP VP

Fig. 3. Architecture of SIGN for sentence matching.

4.2 Assignment of Syntactic Structures

Given a pair of sentences (X, Y), three types of syntactic analysis are respectively applied to X and Y: syntactic dependency parsing, chunking, and POS tagging.

The syntactic dependency parsing outputs a set of directed relations between lexical items in the sentence, for showing their internal structures. Specifically, give a sentence $X = \{w_1, w_2, \dots, w_{t_X}\}$, the dependency parser will output a dependency graph $\mathcal{G}_d^X = (\mathcal{V}_d^X, \mathcal{E}_d^X)$, where $\mathcal{V}_d^X$ contains the words in X, and $\mathcal{E}_d^X = \{(w_i, w_j)\}$, where (w_i, w_j) is a directed edge for indicating that there exists a syntactic dependency between words w_i and w_j. In matching, the head-dependent relations provide an approximation to the semantic relationship between prediction and their arguments, which is useful for matching.

Chunking is another kind of syntactic analysis where the sequence of words in a sentence is identified as forming phrases of various chunk types, such as noun phrases, verb phrases, and prepositional phrases. [1]. Specifically, for a sentence $X = \{w_1, w_2, \dots, w_{t_X}\}$, the chunker will output a chunk sequence $C_X = \{c_1, c_2, \dots, c_n\}$ and its types sequence $\{CHK_{c_1}, CHK_{c_2}, \dots, CHK_{c_n}\}$. Furthermore, from the results, we can obtain $\mathcal{N}_{c_j}$ as well, where $\mathcal{N}_{c_j}$ is the set of word w_i consisting of chunk c_j. In matching, chunking gives additional access to the information about groups of the word, which will give an additional chunk-level matching signal. Therefore, it is efficient for matching hierarchically.

POS and NE tagging are widely useful syntactic analyses where the sequence of words in a sentence is tagged as 11 POS. Specifically, for a sentence $X = \{w_1, w_2, \dots, w_{t_X}\}$, the POS/NER tagger will output a POS tag sequence $\{POS_{w_1}, POS_{w_2}, \dots, POS_{w_{t_X}}\}$ and the NER tag sequence $\{NER_{w_1}, NER_{w_2}, \dots, NER_{w_{t_X}}\}$. For text matching, POS and NE tags are useful for encapsulating the rich structural patterns and good measurements of importance in terms of interactions [6].

4.3 Syntactic-Informed Matching Graph

Based on the sentence pair (X, Y) and their syntactic structures, a directed hierarchical graph $\mathcal{G}_{X,Y} = (\mathcal{V}, \mathcal{E}, \mathbf{H})$ can be derived for matching, referred to as the matching graph in the article.

Specifically, each node in $\mathcal{V}$ could be a word or a chunk in X and Y, or a sentence (i.e., X or Y); $\mathbf{H}$ denotes the matrix that contains the node representations in graph $\mathcal{G}_{X,Y}$, and all of the node representations consist of a matrix $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_{|\mathcal{V}|}] \in \mathbb{R}^{d\times|\mathcal{V}|}$, where each node's representation is denoted as $\mathbf{h}_v \in \mathbb{R}^d, \forall v \in \mathcal{V}$, and $\mathcal{E}$ denotes directed relationships between the nodes.

The nodes in the matching graph are connected with a set of directed edges $\mathcal{E}$, which is defined as the union of several types of edges inferred from relation structures, attribute structures, and aggregation structures:

$$\mathcal{E} = \mathcal{E}_{rel} \cup \mathcal{E}_{att} \cup \mathcal{E}_{agg}.$$

As for the relation structures, the edge $\mathcal{E}_{rel}$ represents that one node syntactic relates to another node, which consists of dependency edge $\mathcal{E}_{dep}$ and chunk edge $\mathcal{E}_{chk}$:

$$\mathcal{E}_{rel} = \mathcal{E}_{dep} \cup \mathcal{E}_{chk}.$$

As for attribute structures, the edge $\mathcal{E}_{att}$ represents that one node has the same attributes as another node, which consists of POS edge $\mathcal{E}_{POS}$, NER edge $\mathcal{E}_{NER}$, and chunk edge $\mathcal{E}_{TOC}$ (**Type of Chunk (TOC)**):

$$\mathcal{E}_{att} = \mathcal{E}_{POS} \cup \mathcal{E}_{TOC} \cup \mathcal{E}_{NER}.$$

To aggregate the syntactic information to a sentence, all of the word nodes and chunk nodes are linked to the corresponding sentence nodes, forming the aggregation edges $\mathcal{E}_{agg}$. The details are described as follows.

4.3.1 Relation Edges. The relation edge $\mathcal{E}_{rel}$ represents that one node syntactic relates to another node, which consists of dependency edge $\mathcal{E}_{dep}$ and chunk edge $\mathcal{E}_{chk}$. Please note that the relation structure interactions of two sentences are mainly referred to as syntactic-based representation methods [8, 9]. The reason for the implementation is that the edge construction time complexity of utilizing fine-grained relation tags (i.e., the syntactic-based interaction method) will become $O(|t_X|^2 \times |t_Y|^2)$. In such an interaction method, it will delay the inference time unbearably compared to the time complexity of representation methods $(O(|t_X| + |t_Y|)^2))$. A more detailed time complexity analysis can be found in Section 4.7.1. At the same time, the space consumption of the interaction method is also not affordable.

Specifically, for the source sentence X in the inputted sentence pair, its syntactic dependency parsing results form a graph $\mathcal{G}_d^X = (\mathcal{V}_d^X, \mathcal{E}_d^X)$, as described in Section 4.2. Similarly, for the target sentence Y, its syntactic dependency parsing results also form another graph $\mathcal{G}_d^Y = (\mathcal{V}_d^Y, \mathcal{E}_d^Y)$. SIGN defines its syntactic dependency edges, denoted as $\mathcal{E}_{dep}$, as the union of these sets:

$$\mathcal{E}_{dep} = \mathcal{E}_d^X \cup \mathcal{E}_d'^X \cup \mathcal{E}_d^Y \cup \mathcal{E}_d'^Y,$$

where $\mathcal{E}_d'^X = \{(w_i, w_j) : (w_j, w_i) \in \mathcal{E}_d^X \vee i = j\}$ and $\mathcal{E}_d'^Y = \{(w_i, w_j) : (w_j, w_i) \in \mathcal{E}_d^Y \vee i = j\}$. Please note that following the practices in the work of Bastings et al. [3], SIGN also includes $\mathcal{E}_d'^X$ and $\mathcal{E}_d'^Y$, which contain the dependency links with inverse directions and self-loop links, as its edges. These edges enable SIGN to propagate the information in the matching graph more efficiently and effectively—that is, they propagate not only along the dependency directions but also along its inverse directions and to itself.

Given a sentence pair (X, Y), the parsed chunking information can also be represented as the edges between the word nodes $\mathcal{V}_w$ and chunk nodes $\mathcal{V}_c$, denoted as $\mathcal{E}_{chk}$:

$$\mathcal{E}_{chk} = \mathcal{E}_{chk}^X \cup \mathcal{E}_{chk}^Y,$$

where $\mathcal{E}_{chk}^X$ (or $\mathcal{E}_{chk}^Y$) is defined as

$$\mathcal{E}_{chk}^X = \{(w_i, c_j) : w_i \in \mathcal{N}_{c_j}, \forall j = 1, 2, \cdots\} \cup \{(c_j, c_j) | c_j \in C_X\},$$

where $\mathcal{N}_{c_j}$ is the set of word w_i consisting of chunk c_j. $\mathcal{E}^Y_{chk}$ is defined similarly. $\mathcal{E}_{chk}$ is used for propagating the information from the word nodes to the corresponding high-level chunk nodes.

4.3.2 Attribute Edges. As for attribute structures, the edge $\mathcal{E}_{att}$ consists of POS edge $\mathcal{E}_{POS}$, NER edge $\mathcal{E}_{NER}$, and chunk edge $\mathcal{E}_{TOC}$. As for the syntactic structure interactions at the attribute level, we adapt the syntactic-based interaction methods [6, 25] for the following two reasons: (1) previous studies [54] showed that the fine-grained matching signals will be more robust and accurate, and (2) the time complexity of constructing attribute edges ($O(|t_X| \times |t_Y|)$) is affordable.

The constructing detail is explained as follows. It is a strong signal in matching if two matched words (or chunk) have identical POS tags (or chunk type labels). Inspired by the observation, SIGN defines the POS edges (denoted as $\mathcal{E}_{POS}$) as follows:

$$\mathcal{E}_{POS} = \{(w_i, w_j) : w_i \in X \wedge w_j \in Y \wedge POS_{w_i} = POS_{w_j}\} \cup \{(w_i, w_j) : w_i \in Y \wedge w_j \in X \wedge POS_{w_i} = POS_{w_j}\},$$

where POS_{w_i} and POS_{w_j} are the assigned POS tags for word w_i in X and word w_j in Y.

Similarly, the NER edges (denoted as $\mathcal{E}_{NER}$) are defined as follows:

$$\mathcal{E}_{NER} = \{(w_i, w_j) : w_i \in X \wedge w_j \in Y \wedge NER_{w_i} = NER_{w_j}\} \cup \{(w_i, w_j) : w_i \in Y \wedge w_j \in X \wedge NER_{w_i} = NER_{w_j}\}.$$

At the chunk level, SIGN defines the TOC edges (denoted as $\mathcal{E}_{TOC}$) as follows:

$$\mathcal{E}_{TOC} = \{(c_i, c_j) : c_i \in C_X \wedge c_j \in C_Y \wedge CHK_{c_i} = CHK_{c_j}\} \cup \{(c_i, c_j) : c_i \in C_Y \wedge c_j \in C_X \wedge CHK_{c_i} = CHK_{c_j}\},$$

where CHK_{c_i} and CHK_{c_j} are the assigned chunk types for chunk c_i from C_X and chunk c_j from C_Y, respectively.

Please note that in the matching task, we assume that the words and chunk from X and Y should be aligned with each other. Therefore, if there is a POS/NER edge (or a TOC edge) from w_i to w_j (or c_i to c_j), there will also be an edge from w_j to w_i (or c_j to c_i) so that the information can be propagated along both directions.

4.3.3 Aggregation Edges. Finally, all of the word nodes and chunk nodes are linked to the corresponding sentence nodes, forming the aggregation edges:

$$\mathcal{E}_{agg} = \mathcal{E}^X_{agg} \cup \mathcal{E}^Y_{agg},$$

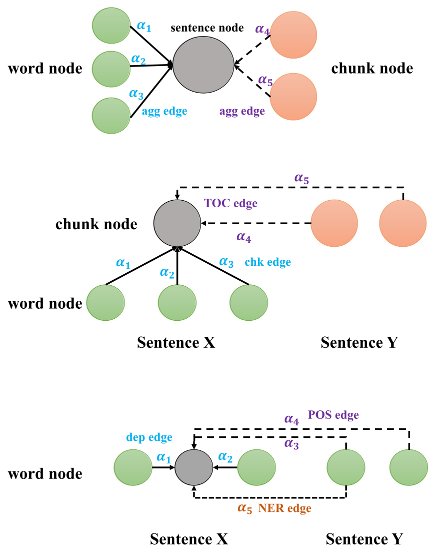
where $\mathcal{E}^X_{agg} = \{(w_i, x)|w_i \in X\} \cup \{(c_j, x)|c_j \in C_X\} \cup \{(x, x)\}$ and $\mathcal{E}^Y_{agg} = \{(w_i, y)|w_i \in Y\} \cup \{(c_j, y)|c_j \in C_Y\} \cup \{(y, y)\}$. These links aggregate the information from the words and chunks to the corresponding sentence nodes.

The details are described in Section 1 of the supplementary material. Note that any NLP parser can be used for the construction of the edges. In this article, we used the Stanford CoreNLP parser [31].

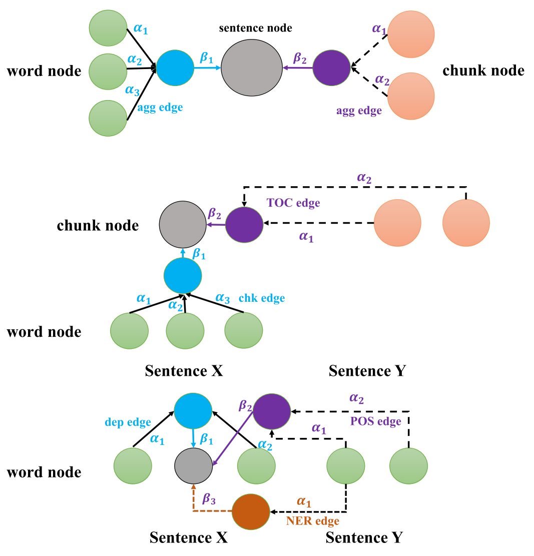
4.4 Initialization of Graph Representations

In SIGN, the matching between X and Y is formalized as representation learning over the matching graph $\mathcal{G}_{X,Y} = (\mathcal{V}, \mathcal{E}, \mathbf{H})$. In graph representation learning, the node representations are updated at each layer. We denote the node representation matrix through the k-th layer ($k = 0, 1, 2 \cdots$) message propagating as $\mathbf{H}^{(k)}$.

The initial representations of the nodes, denoted as $\mathbf{H}^{(0)} = [\mathbf{h}^{(0)}_1, \ldots, \mathbf{h}^{(0)}_{|\mathcal{V}|}]$, are given by the embedding vectors obtained from a PLM. Specifically, after inputting sentence pair (X, Y) into the PLM with format $H_{PLM} = \{[CLS], w^X_1, \ldots, w^X_{t_X}, [SEP]\}, w^Y_1, \ldots, w^Y_{t_Y}, [SEP]$, we can obtain the embeddings from the last hidden layer of the PLM model: $\mathbf{h}_{PLM} = \{\mathbf{h}_v, \forall v \in H_{PLM}\}$. The initial representations of the word nodes in matching graph $\mathbf{H}^{(0)}$ are set with the corresponding vectors $\mathbf{h}_{PLM}$'s.



(a) GAT algorithm in SIGN



(b) HGAT algorithm in SIGN

Fig. 4. The algorithm derived in representation learning over the matching graph. The top, middle, and bottom subfigures represent the syntactic edges at the word, chunk, and sentence levels, respectively. (a) GAT, where α_i is the attention weight of different input-link edges (i.e., $\sum_i \alpha_i = 1$). (b) HGAT, where α_i is the attention weight of single edge types (i.e., $\sum_i \alpha_i = 1$) and β_j is the attention weight of different edge types (i.e., $\sum_j \beta_j = 1$).

As for the chunk nodes in $\mathcal{V}_c$, we follow the practices in the work of Chen et al. [7] to set the initial embeddings based on the embeddings of the associated words utilizing a gating mechanism: $\mathbf{h}_{c_j} = \sum_{w_i \in \mathcal{N}_{c_j}} \mathrm{FFN}_c(\mathbf{h}_{w_i}; \Theta^c) \odot \mathbf{h}_{w_i}$, for all $h_{c_j} \in \mathcal{V}_c$, where FFN_c is a two-layer feed-forward neural network with parameters Θ^c and '$\odot$' denotes the element-wise product of two vectors. Here FFN_c plays as a gating function. In a sentence, the semantic contributions of the functional words (e.g., "the," "a") to the whole sentence are relatively small. In the gating functions, the words with richer semantics in the chunk can contribute more to the chunk representation.

Finally, for initializing the sentence nodes, the embeddings of all associated words are aggregated with max-pooling: $\mathbf{h}_s = \text{max-pooling } \mathbf{h}_{w_i}$, for all $s \in \mathcal{V}_s$.

4.5 Algorithm Derived in Representation Learning over the Matching Graph

In SIGN, the GAT is applied to learn the node semantic representation incorporating the edge syntactic information. We applied GAT [45] and HGAT [47]. Figure 4 shows the intuitive example of the GAT and HGAT algorithms.

In the GAT algorithm, different types of edge have the same message passing mechanism, which denotes that any input-link edge has attention weight α_i (i.e., $\sum_i \alpha_i = 1$). Figure 4(a) gives an intuitive example: in the bottom subfigure, the word node receives the message from the dependency edges with weight α_1, α_2, POS edges with weight α_3, α_4, and NER edges with weight α_5, the chunk nodes, and the sentence node, which also receives a similar message from its linked edges.

The HGAT algorithm introduced different message passing mechanisms for different types of edge. Figure 4(b) gives an intuitive example: in the bottom subfigure, first, each edge type receives the message with α_i weight (i.e., $\sum_i \alpha_i = 1$) from its linked node. For example, the dependency type receives the message with weight α_1, α_2. Next, different edge types pass the message to the target node with weight β_j (i.e., $\sum_j \beta_j = 1$). For example, each word node receives the message from the dependency, POS, and NER edge with weight $\beta_1, \beta_2, \beta_3$. Similarly, the chunk node receives the message from the chunk and TOC edges, and the sentence node receives the aggregate edges.

Next, we explain their message passing mechanisms in detail.

4.5.1 GAT Representation Learning. The initial representations in $\mathbf{H}^{(0)}$ only contain the local semantic information of the sentence elements. The GAT [45] is employed to learn the node representations in the matching graph so that the node information can be propagated along the edges. For each depth $\ell = 0, \ldots, L$, where L is the maximal number of layers, the GAT generates a neighborhood embedding with the aggregation function for each node and combines it with the existing embedding of the node. To avoid a vanishing gradient problem and following the mechanism in the work of Devlin et al. [11], the node representations are updated as follows:

$$\mathbf{H}^{(\ell+1)} = \text{LayerNorm}(\mathbf{H}^{'(\ell)} + \mathbf{H}^{(\ell)} + \text{FFN}_\ell(\mathbf{H}^{'(\ell)} + \mathbf{H}^{(\ell)}; \Theta^\ell)), \tag{1}$$

where FFN_ℓ is a one-layer MLP, parameterized by Θ^ℓ. $\mathbf{H}^{'(l)} = [\mathbf{h}_1^{'(\ell)}, \ldots, \mathbf{h}_{|\mathcal{V}|}^{'(\ell)}]$ and its i-th column $\mathbf{h}_i^{'(\ell)}$ is defined as

$$\mathbf{h}_i^{'(\ell)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j^{(\ell)}\right),$$

for $\ell = 0, \ldots, L$ where K is the number of headers in the attention module, $\mathcal{N}_i$ denotes the set of embeddings of the neighborhood with the nodes corresponding to h_i, α_{ij}^k represents the normalized attention coefficients computed by the k-th attention mechanism α^k, and $\mathbf{W}^k$ is the corresponding input linear transformation's weight matrix. The α_{ij}^k is defined as

$$\alpha_{ij}^k = \frac{\exp\left(\text{LeakyRelu}(\mathbf{a}^T[\mathbf{W}^k\mathbf{h}_i \parallel \mathbf{W}^k\mathbf{h}_j])\right)}{\sum_{j\in N_i}\exp(\text{LeakyRelu}(\mathbf{a}^T[\mathbf{W}^k\mathbf{h}_i \parallel \mathbf{W}^k\mathbf{h}_j]))},$$

where '$\|$' denotes the concatenation operation. In our experiments, the attention mechanism is a single-layer feed-forward neural network, parameterized by a weight vector $\mathbf{a}^T$.

4.5.2 HGAT Representation Learning. The HGAT [47] provided a better way to leverage different types of syntactic structures in the graph learning phase. Since our matching graph is heterogeneous in different syntactic structures, the HGAT is employed to learn the different types of node representations in the matching graph so that the node information can be propagated along with the different types of edges. For each depth $\ell = 0, \ldots, L$, where L is the maximal number of layers and for each type of edges $e_i (i = 1, 2, \ldots, r_a + r_n + 1)$ described in Section 4.3, the HGAT generates different types of node representations. Then the HGAT applies the aggregation function for each type of node representation of its neighbors and combines all types of representations with a learnable aggregating function. To incorporate the graph learning algorithm and the PLM fine-tuning, we adapt the skip connect [11] mechanism. The node representations are updated as follows:

$$\mathbf{H}^{(\ell+1)} = \text{LayerNorm}(\mathbf{H}^{'(\ell)} + \mathbf{H}^{(\ell)} + \text{FFN}_\ell(\mathbf{H}^{'(\ell)} + \mathbf{H}^{(\ell)}; \Theta^\ell)), \tag{2}$$

where FFN_ℓ is a one-layer MLP, parameterized by Θ^ℓ. $\mathbf{H}^{'(l)} = [\mathbf{h}_1^{'(\ell)}, \ldots, \mathbf{h}_{|\mathcal{V}|}^{'(\ell)}]$ and its i-th column $\mathbf{h}_i^{'(\ell)}$ is computed through semantic level attention:

$$\mathbf{h}_i^{'(\ell)} = \sum_{j=1}^{r_a+r_n+1} \beta_{e_j}\mathbf{h}_{i,e_j}^{'(\ell)}, \tag{3}$$

where $\mathbf{h}_{i,e_j}^{'(\ell)}$ is the representation updated by $\mathbf{h}_i^{(\ell)}$ through edge e_j, β_{e_j} denotes the coefficient of the $\mathbf{h}_{i,e_j}^{'(\ell)}$, and β_{e_j} can be updated as follows:

$$\beta_{e_j} = \frac{exp(w_{e_j})}{\sum_{k=1}^{N_c} exp(w_{e_k})},$$

where w_{e_j} denotes the importance of the type e_j parameterized by W^e, and it can be updated as follows:

$$w_{e_j} = \frac{1}{|\mathcal{V}|} \sum_{i \in |\mathcal{V}|} \left(h_i^{(\ell)}\right)^T \tanh\left(W^e \mathbf{h}_{i,e_j}^{'(\ell)}\right).$$

As for $\mathbf{h}_{i,e_j}^{'(\ell)}$, the node level attention is applied with it:

$$\mathbf{h}_{i,e_j}^{'(\ell)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{m \in \mathcal{N}_i} \alpha_{im,e_j}^k \mathbf{W}_{e_j}^k \mathbf{h}_{m,e_j}^{(\ell)}\right), \tag{4}$$

for $\ell = 0, \ldots, L$, where K is the number of headers in the attention module, $\mathcal{N}_i$ denotes the set of embeddings of the neighborhood with the nodes corresponding to h_i, α_{im,e_j}^k is the normalized attention coefficients computed by the k-th attention mechanism $\alpha_{e_j}^k$, and $\mathbf{W}_{e_j}^k$ is the corresponding input linear transformation's weight matrix. The α_{im,e_j}^k is defined as

$$\alpha_{im,e_j}^k = \frac{\exp\left(\text{LeakyRelu}(\mathbf{a}^T[\mathbf{W}_{e_j}^k \mathbf{h}_{i,e_j} \parallel \mathbf{W}_{e_j}^k \mathbf{h}_{m,e_j}])\right)}{\sum_{m \in N_i} \exp(\text{LeakyRelu}(\mathbf{a}^T[\mathbf{W}_{e_j}^k \mathbf{h}_{i,e_j} \parallel \mathbf{W}_{e_j}^k \mathbf{h}_{m,e_j}]))},$$

where '‖' denotes the concatenation operation. In our experiments, the attention mechanism is a single-layer feed-forward neural network, parameterized by a weight vector $\mathbf{a}^T$.

Note that the syntactic-informed graph is actually a hierarchical graph, which consists of the sentence, chunk, and word level. In such a graph, the way of aggregating embeddings from word/chunk level to chunk/sentence level can be well incorporated into the graph attention mechanism. For example, the set of attention weights $\{\alpha_{im,e_j}^k | e_j \in \mathcal{E}_{agg}\}$ can be seen as the aggregation weights from the word/chunk level to the sentence level. The set of attention weights $\{\alpha_{im,e_j}^k | e_j \in \mathcal{E}_{chk}\}$ can be seen as the aggregation weights from the word level to the chunk level.

4.6 Relation Classifier

To capture different semantics on different types of nodes, the node embeddings are aggregated as features. For sentence X in the inputted pair (X, Y), its aggregated representation, denoted as $\mathbf{g}^X$, is obtained by aggregating the representations of associated nodes:

$$\mathbf{g}^X = \underset{w \in \mathcal{V}_w \wedge w \in X}{\text{max-pooling}}\, \mathbf{h}_w^L \parallel \underset{c \in \mathcal{V}_c \wedge c \in X}{\text{max-pooling}}\, \mathbf{h}_c^L \parallel \mathbf{h}_x^L,$$

where $\mathbf{h}_w^L$ and $\mathbf{h}_c^L$ are the word and chunk representations associated with word w and chunk c at the L-th level. Similarly, the aggregated representation of Y, denoted as $\mathbf{g}^Y$, can also be calculated based on the same way.

Therefore, the final matching can be predicted by summarizing the matching signals in $\mathbf{g}^X$, $\mathbf{g}^Y$, and the $\mathbf{h}_{[CLS]}$ vector outputted from BERT:

$$\mathbf{p}(X, Y) = \text{FFN}_p\left(\left[\mathbf{g}^X \parallel \mathbf{g}^Y \parallel \mathbf{g}^X \odot \mathbf{g}^Y \parallel |\mathbf{g}^X - \mathbf{g}^Y|\right] + \mathbf{h}_{[CLS]}; \Theta^p\right), \tag{5}$$

where $\mathbf{p}(X, Y) = [p_1, \ldots, p_{|\mathcal{Z}|}]$ and p_k denotes the probability of the k-th category. FFN_p is an MLP parametered by Θ^p. The last layer of FFN_p is softmax so that its output is a probability distribution.

The SIGN model has free parameters $\Theta = \{\Theta^c, \Theta^\ell, \Theta^p, \mathbf{W}^k, \mathbf{a}, \mathbf{W}^e\}$ to determine. In the training phase, given the training set $\mathcal{D} = \{(X_i, Y_i, z_i)\}_{i=1}^N$, the parameters are learned by minimizing the cross-entropy loss between the labels and the predicted results:

$$\mathcal{L} = -\sum_{(X,Y,\mathbf{z})\in\mathcal{D}} \sum_{k=1}^{|\mathcal{Z}|} z_k \log p_k + \lambda\|\Theta\|^2, \tag{6}$$

where $z_k = 1$ when the pair belongs to the k-th category, and $z_k = 0$ in other cases, $\|\Theta\|^2$ is the ℓ_2 regularizer for avoiding overfitting, and $\lambda > 0$ is the tradeoff coefficient.

4.7 Discussion

SIGN provides a general and elegant graph learning approach to sentence matching. Furthermore, SIGN can match two sentences efficiently and effectively.

4.7.1 Time Complexity Analysis. At the online time, SIGN needs to process the sentence pairs with the PLM, parse them with the NLP parser, solve the GAT, and finally calculate the matching score. The online time complexity for a typical PLM [11, 28] and NLP parser [31] is of $\mathcal{O}(|t_X + t_Y|^2 \times d)$ and $\mathcal{O}((|t_X|^2 + |t_Y|^2) \times d)$, where d is the embedding dim of each word. Since the NLP parser can work with the PLM extraction in parallel, usually the NLP parser does not delay the inference time.

At the online time, the time complexity of the relaxed SIGN is corresponding to the time complexity of the GAT, which is of $\mathcal{O}((m + n^2) \times d)$ [51] on the association graph, where n is the total number of nodes and m is the total number of edges. We can see, in practice, that the edge number m is nearly close to the nodes n (i.e., $m \approx n$). In this way, the total time complexity of SIGN is $\mathcal{O}((|t_X + t_Y|^2 + (|t_X|^2 + |t_Y|^2) + m + n^2) \times d) = \mathcal{O}(|t_X + t_Y|^2 \times d)$ with the PLM as the inference time bottleneck.

4.7.2 Syntax and Semantics Incorporation. The graph learning framework provides a powerful tool to model different types of heterogeneous syntactic structures. More importantly, both the word semantics and the syntactic structures are jointly represented in one graph: the semantic information is encoded as the node embeddings, and the syntactic structures are represented as the directed edges. In this way, SIGN provides a unified approach to enable the two matching sentences to interact in a fine-grained manner, enhancing the overall understanding and coherence of the context. The matching graph models all information from two sentences into one graph so that their interactions from semantic and syntactic perspectives can be fully exploited.

SIGN also conducts sentence matching in an explainable way. We found that the word-level and chunk-level similarities generated by SIGN are sparse. The multiple types of syntactic information not only improve the overall matching accuracy but also help to discriminate the word/chunk pairs that are either dissimilar or similar in form but different in syntax. Therefore, SIGN is easier to explain than existing methods like BERT and RoBERTa.

5 EXPERIMENTS

In this section, we evaluate the performance of the proposed SIGN model with experiments on three publicly available datasets. We raised and tried to answer three major research questions:

- *RQ1*: Can SIGN improve sentence matching?
- *RQ2*: What are the benefits of using different types of syntactic structures in matching?
- *RQ3*: How does SIGN improve the matching accuracy in an explainable way?

The source code and all experiments have been shared on GitHub (https://github.com/XuChen0427/Syntactic-Informed-Graph-Networks-for-Sentence-Matching).

Table 3. Statistics of Three Datasets Used in the Experiments

Dataset	Task Description	$\lvert\mathcal{Z}\rvert$	# Pairs
QQP	paraphrase identification	2	404k
SNLI	premise-hypothesis prediction	3	570k
SciTail	premise-hypothesis prediction	2	27k

5.1 Experimental Settings

We conducted the experiments to test the performance of SIGN using three large-scale publicly available sentence matching benchmark datasets: QQP for the task of PI, and **Stanford Natural Language Inference (SNLI)** [4] and SciTail [20] for the task of NLI (Table 3 summarizes the description and statistics of them):

QQP[1] is a dataset for PI. The label has two classes indicating whether one question is a paraphrase of the other. The dataset contains 404k labeled sentence pairs. We used the same data split as in the work of Wang et al. [50].

SNLI[2] is a benchmark dataset for NLI. The dataset contains 570k labeled sentence pairs. In NLI, each input triple represents premise-hypothesis-label, and the label could be "entailment," "neutral," "contradiction," or "-". Following the practices in the work of Bowman et al. [4], we used the same dataset split, and the sentence pairs labeled as "-" were ignored.

SciTail[3] is another entailment dataset based on multiple-choice science exams and web sentences. Each input data triple also represents premise-hypothesis-label. The label is "entailment" or "neutral" because scientific factors cannot contradict. The dataset contains 27k labeled sentence pairs.

Several sentence-matching state-of-the-art baselines that did not utilize syntactic structures were chosen as the baselines, including BIMPM [49], CSRAN [21], RE2 [55], BERT [11], RoBERTa [28]:

BIMPM [49] conducts multiple-perspective matching so that one sentence in one timestep is matched against all timesteps of another sentence.

CSRAN [21] performs multi-level attention between two sentences with residual connections among multiple levels.

RE2 [55] introduces an architecture based on various augmented residual connections between convolutional layers and attention layers for short text matching.

BERT [11] and *RoBERTa* [28] append an MLP on the last hidden state corresponding to the first token "[CLS]" of sequence output and fine-tunes the weights on the task.

At the same time, we compared some other baselines: pt-DecAttn [44], DIIN [15],MwAN [40], SAN [26], DRCN [43], and DGEM [20].

SIGN was also compared with the baselines that utilize syntactic structures:

HIM [9] is a model that uses the constituency tree to improve local word representation.

TBCNN [34] incorporates the syntax trees and utilizes a CNN-based aggregation function to improve the NLI tasks with the representation-based methods.

SyntaxBERT [2] incorporates the syntax trees into pre-trained Transformers to improve sentence representation quality. State-of-the-art performance was achieved on glue tasks.

SS-BERT [25] incorporates the syntax trees and semantic role labeling into sentence matching with the representation-based methods.

Table 4. Types of Chunk and Chunking Rules

Type of Chunk	**Chunking Rule**
NP	{<DT|PR.*| JJ.*|NN.*>+}
PP	{<IN><NP>}
VP	{<VB.*><VB.*>+} ∪ {<VB.*><NP>+}
CLAUSE	{<NP><VBP>}
The same as POS of word	If only contain one word

The angle brackets (<>) represent the sequence of the POS, the asterisk (*) represents the wildcard character, the vertical line (|) represents a matched tag number (e.g., the number of DT) ≥ 0, and the plus sign (+) represents a matched tag number ≥ 1.

ConSeqNet [48] utilized other knowledge from knowledge graphs (e.g., ConceptNet, WordNet) to improve the performance of NLI tasks.

These syntactic-based baselines only utilized limited syntactic structures and encoded different syntax *separately*. Our model SIGN can leverage multiple types of syntactic structures in a syntactic graph and make them interact in a fine-grained way. Some results are not available on all three datasets because the implementations are not publicly available.

To get the syntactic structure information, Stanford CoreNLP [31] was used to tokenize the inputted sentences, generate the syntactic dependency trees, and predict the POS tags, NER tags, and chunk types. Please note that although the PLM was used for generating the initial embeddings of words, here we still adopted the Stanford CoreNLP toolkit rather than the pre-trained language tokenizer [11, 28] for tokenization. To avoid over-sparse edges in the matching graph, in all of the experiments, we configured the Stanford CoreNLP tool so that all outputted POS tags are from $\{DT, IN, JJ, NN, PR, RB, TO, VB, WP, WR, UNK\}$ and all outputted chunk types are from $\{NP, PP, VP, CLAUSE\}$. The detailed rules are shown in Table 4. It would be a future research direction to reduce the noise of confusing parsing results.

In all of the experiments, we did not limit the maximum sequence length, and all sequences in a batch were padded to the batch-wise maximum of sequence length. In the training process, all of the models were trained using the Adam optimizer [22], with the learning rate η tuned among $[1e-5, 3e-5]$ and the batch size n tuned as in other works [16, 32]. In the graph representation learning layer and the prediction layer, the dropout with a keep probability of 0.8 was adopted. In the PLM, the keep probability was set to 0.9. The number of GAT/HGAT layers was tuned in the range of 1 to 3. The threshold for gradient clipping was set to 5.

5.2 Evaluation and Analysis

We conducted experiments and analysis to answer the aforementioned three research questions.

5.2.1 *RQ1: Can SIGN Improve Sentence Matching?*

Tables 5, 6, and 7 report the experimental results on the benchmark datasets of QQP, SNLI, and SciTail, respectively. Following the practices in other works [15, 49, 55], prediction accuracy was used as the evaluation metric. We report the derivations of SIGN with different PLMs and different graph learning algorithms—for example, "SIGN-BERT$_{BASE}$-GAT" denotes the BERT$_{BASE}$ PLM and GAT algorithm described previously. We can see that the proposed SIGN model outperformed all of the baselines on all of the three datasets, indicating the effectiveness of using syntactic structures in sentence matching. The experimental results are not surprising, as SIGN used not only the syntactic structures of the sentences but also the semantic representations from the PLM to initialize the matching graph. The results also indicate that given the strong results of BERT$_{BASE}$, BERT$_{LARGE}$, and RoBERTa$_{LARGE}$

Table 5. Performance Comparison on the QQP Test Set

Without Syntactic Structures	**Acc. (%)**
pt-DecAttn-word [44]	87.5
pt-DecAttn-char [44]	88.4
DIIN [15]	89.1
MwAN [40]	89.1
CSRAN [21]	89.2
SAN [26]	89.4
RE2 [55]	89.2
$\text{BERT}_{BASE}{}^{\dagger}$ [11]	89.4
$\text{BERT}_{LARGE}{}^{\dagger}$ [11]	89.6
$\text{RoBERTa}_{LARGE}{}^{\dagger}$ [28]	90.0
Using Syntactic Structures	**Acc. (%)**
$\text{HIM}^{\dagger}$ [9]	88.7
$\text{SS-BERT}_{BASE}{}^{\dagger}$ [25]	89.4
$\text{SS-BERT}_{LARGE}{}^{\dagger}$ [25]	89.7
SyntaxBERT_{BASE} [2]	89.6
$\text{SyntaxBERT}_{LARGE}$ [2]	89.5
Ours (SIGN-BERT$_{BASE}$-GAT)†	**89.9***
Ours (SIGN-BERT$_{LARGE}$-GAT)†	**90.0***
Ours (SIGN-RoBERTa$_{LARGE}$-GAT)†	**90.5***
Ours (SIGN-BERT$_{BASE}$-HGAT)†	**90.1***
Ours (SIGN-BERT$_{LARGE}$-HGAT)†	**90.2***
Ours (SIGN-RoBERTa$_{LARGE}$-HGAT)†	**90.9***

The asterisk (*) represents that the improvements over the corresponding PLM baseline are statistically significant (t-test and p-value < 0.05), and the dagger (†) denotes that the models are our implementations and are trained with the same settings.

(the best baseline), SIGN can further improve the results by leveraging the explicit syntactic information.

We also note that SIGN outperformed the baselines of HIM [9], TBCNN [34], SS-BERT [25], and SyntaxBERT [2], which also used the syntactic structures for matching. Comparing the model structures of HIM, TBCNN, and SIGN, we found that HIM and TBCNN use the syntactic dependency structures of the two sentences to respectively enhance their representations. SIGN, however, constructs one syntactic-informed matching graph based on two sentences. Comparing SS-BERT, SyntaxBERT, and SIGN, they all utilized the PLM and syntactic structures to conduct sentence matching. SIGN, however, utilized different syntactic structures to conduct matching. The results clearly demonstrate that the syntactic-informed matching graph has the advantage of fully exploiting the rich syntactic and semantic information for sentence matching.

5.2.2 *RQ2: What Are the Benefits of Using Different Types of Syntactic Structures in Matching?*

We also conducted two ablation experiments to investigate the benefits of different types of syntactic structures, using the results of the QQP, SNLI, and SciTail on SIGN-BERT$_{BASE}$-GAT and SIGN-BERT$_{BASE}$-HGAT. We respectively investigated for the edges inferred from the relation structure, attribute structure, and aggregation structure. Moreover, we investigated the edges inferred from the syntactic dependency and word-chunk relation structure and POS, NER, and chunk-type attribute structure. Similar phenomenons have also been observed in the experiments on other PLMs.

Table 6. Performance Comparison on the SNLI Test Set

Without Syntactic Structures	**Acc. (%)**
ESIM [9]	88.0
DIIN [15]	88.0
MwAN [40]	88.3
CAFE [42]	88.5
SAN [26]	88.6
CSRAN [21]	88.7
DRCN [43]	88.9
RE2 [55]	89.0
$\text{BERT}_{BASE}{}^{\dagger}$ [11]	89.0
$\text{BERT}_{LARGE}{}^{\dagger}$ [11]	89.2
$\text{RoBERTa}_{LARGE}{}^{\dagger}$ [28]	90.1
Using Syntactic Structures	**Acc. (%)**
TBCNN [34]	83.5
HIM [9]†	88.6
$\text{SS-BERT}_{BASE}{}^{\dagger}$ [25]	89.2
$\text{SS-BERT}_{LARGE}{}^{\dagger}$ [25]	89.7
SyntaxBERT_{BASE} [2]	87.8
$\text{SyntaxBERT}_{LARGE}$ [2]	89.0
Ours (SIGN-BERT$_{BASE}$-GAT)†	**89.5**
Ours (SIGN-BERT$_{LARGE}$-GAT)†	**89.8**
Ours (SIGN-RoBERTa$_{LARGE}$-GAT)†	**90.4***
Ours (SIGN-BERT$_{BASE}$-HGAT)†	**89.7**
Ours (SIGN-BERT$_{LARGE}$-HGAT)†	**89.9***
Ours (SIGN-RoBERTa$_{LARGE}$-HGAT)†	**90.7***

The asterisk (*) represents that the improvements over the corresponding PLM baseline are statisticaly significant (t-test and p-value < 0.05), and the dagger (†) denotes that the models are our implementations and are trained with the same settings.

All experiments were conducted five times, and the averaged accuracy was reported. We conducted significant testing, and all improvements are significant (t-test and p-value < 0.05).

First, we conducted ablation experiments on the QQP and SNLI datasets to test how SIGN-GAT performed if a specific type of edges (inferred from the corresponding type of syntactic structure) were removed from the matching graph. Figures 5 and 6 illustrate the accuracy of these SIGN variations on the test data. The matching accuracy of the original SIGN model (denoted as "all edges") and the best baseline $\text{BERT}_{\text{BASE}}$ are also reported.

Comparing the original version of SIGN-GAT with its variations, we can see that the matching accuracy dropped if any type of the edges were removed. The most performance drops were caused by removing the dependency edges and POS edges. The results indicate that although the syntactic structures of dependency and POS are heterogeneous, both of them are essentially important in sentence matching. The results also verified the effectiveness of encoding different types of syntactic structures as the graph edges. Further note that $\text{BERT}_{\text{BASE}}$ did not use any syntactic information, resulting in the worst performances in Figures 5 and 6. The results showed that the different types of edges are complementary in sentence matching. They can improve the accuracy from different perspectives.

Table 7. Performance Comparison on the SciTail Test Set

Without Syntactic Structures	**Acc. (%)**
DGEM [20]	77.3
CAFE [42]	83.3
CSRAN [21]	86.7
RE2 [55]	86.6
BERT_{BASE}[†] [11]	89.5
BERT_{LARGE}[†] [11]	90.6
RoBERTa_{LARGE}[†] [28]	91.5
Using Syntactic Structures	**Acc. (%)**
HIM [9][†]	71.6
ConSeqNet [48]	85.2
SS-BERT_{BASE}[†] [25]	89.6
SS-BERT_{LARGE}[†] [25]	91.0
Ours (SIGN-BERT$_{BASE}$-GAT)[†]	**90.3**
Ours (SIGN-BERT$_{LARGE}$-GAT)[†]	**91.2**[*]
Ours (SIGN-RoBERTa$_{LARGE}$-GAT)[†]	**92.3**[*]
Ours (SIGN-BERT$_{BASE}$-HGAT)[†]	**90.6**
Ours (SIGN-BERT$_{LARGE}$-HGAT)[†]	**91.7**[*]
Ours (SIGN-RoBERTa$_{LARGE}$-HGAT)[†]	**92.9**[*]

The asterisk (*) represents that the improvements over the corresponding PLM baseline are statistically significant (t-test and p-value < 0.05), and the dagger (†) denotes that the models are our implementations and are trained with the same settings.

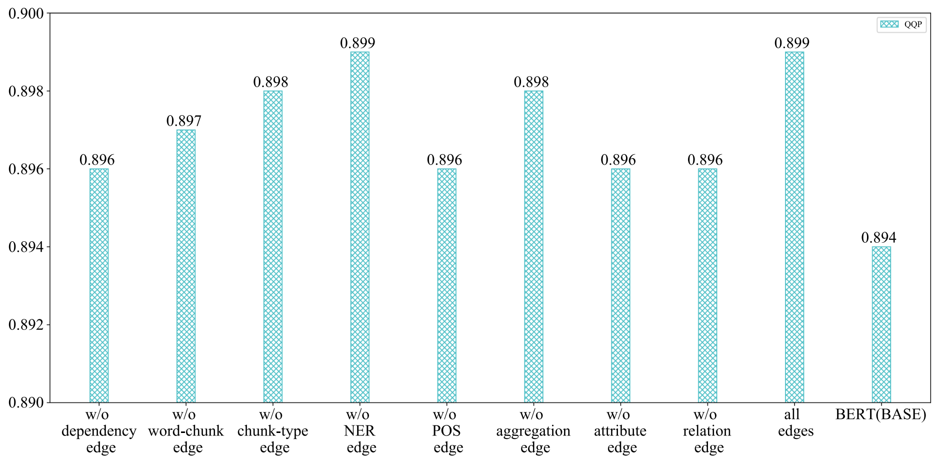


Fig. 5. Ablation studies of the edges inferred from different types of syntactic structures for "SIGN-BERT$_{BASE}$-GAT." The experiment was conducted based on the QQP dataset.

Second, we conducted ablation experiments to test how SIGN-HGAT performed if a specific type of edges (inferred from the corresponding type of syntactic structure) were removed from the matching graph. Table 8 illustrates the accuracy of these SIGN-HGAT variations on the test data. The matching accuracy of the original SIGN model (denoted as "all edges") and the best

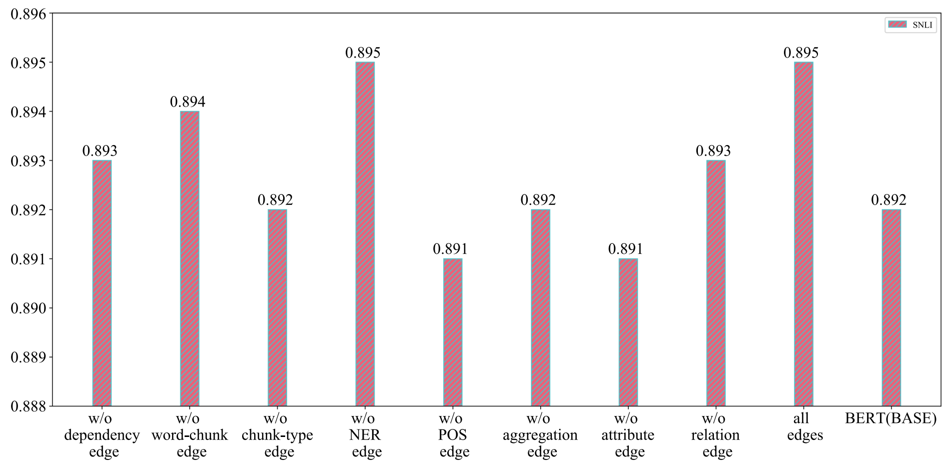


Fig. 6. Ablation studies of the edges inferred from different types of syntactic structures for "SIGN-BERT$_{BASE}$-GAT." The experiment was conducted based on the SNLI dataset.

Table 8. Ablation Study of the Heterogeneous Graph Attention Network Based on PLM BERT$_{BASE}$ on the SciTail Test Set

Ablation Study Model	**SciTail:Acc. (%)**
PLM(BERT$_{BASE}$) [11]	89.5
SIGN-HGAT-w/o relation structure	90.2 (±0.11)
SIGN-HGAT-w/o attribute structure	90.0 (±0.22)
SIGN-HGAT-w/o aggregation structure	90.4 (±0.15)
SIGN-HGAT-w/o syntactic dependency relation	90.3 (±0.14)
SIGN-HGAT-w/o word-chunk relation	90.2 (±0.12)
SIGN-HGAT-w/o chunk-type attribute	90.2 (±0.25)
SIGN-HGAT-w/o POS attribute	90.1 (±0.19)
SIGN-HGAT-w/o NER attribute	90.4 (±0.18)
SIGN-HGAT-all edges	**90.6** (±0.18)

baseline BERT$_{BASE}$ are also reported. Comparing the original version of SIGN with its variations, we can see that the matching accuracy dropped if any type of the edges were removed. The most performance drops were caused by removal of attribute edges—that is, the chunk-type edges and the POS tag edges.

We also conducted experiments for heterogeneous average attention weight β_{e_j} for different syntactic structures for SIGN-HGAT on the SciTail training set. β_{e_j} reflects the importance of each syntactic structure during training. Attention weights β_{e_j} in Figure 7 show that each structure has a certain effect on matching (e.g., the lowest attention weight (NER attribute) is 0.128). Moreover, Figure 7 shows that attribute structures, especially POS and chunk-type attributes, play a more important role than other syntactic structures.

Note that although we can see that different types of syntactic structures help conduct sentence matching, the noise of syntactic structures still hinders the potential of SIGN. We analyzed the bad cases and found that some pairs contain ungrammatical sentences—for example, sentences without subjects or verbs (e.g., the pair ("white prisms"; "a prism helps people understand") in the Scitail test set). These sentences may result in unreasonable syntactic parsing results. It is difficult

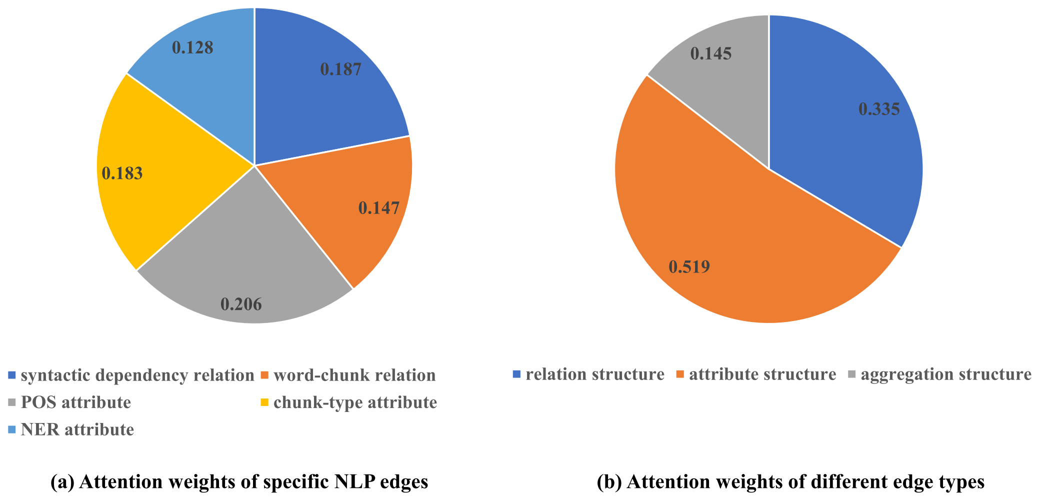


Fig. 7. Average attention weight β_{e_j} inferred from different syntactic structure on the SciTail training set.

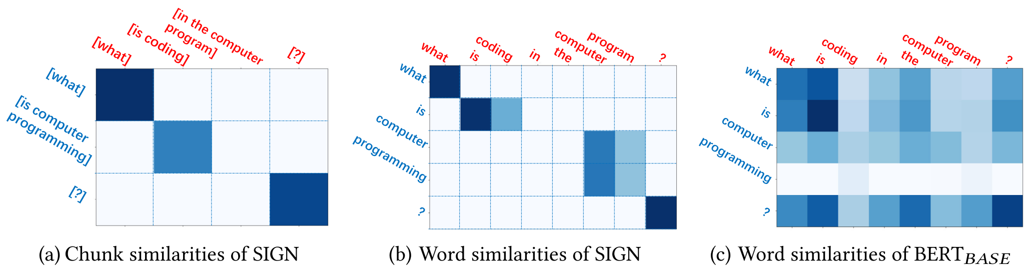


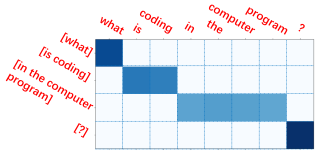
Fig. 8. Cross-sentence chunk and word similarities for a sentence pair. Darker means a higher value. The SIGN and BERT$_{BASE}$ models are trained on the QQP dataset.

for our method to handle them. How to deal with some ungrammatical sentences in real matching practices will be future work.

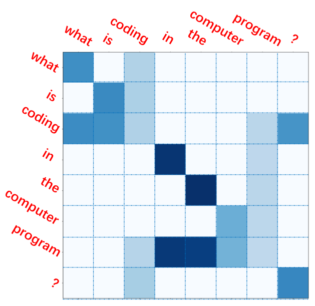
5.3 Explainability Analysis

In this section, we aim to answer RQ3: How does SIGN improve the matching accuracy in an explainable way? We investigated how SIGN improves the matching accuracy in an explainable way with its best variation "SIGN-RoBERTa$_{LARGE}$-HGAT" improving the matching accuracy, using the example (X = "*What is computer programming ?*"; Y = "*What is coding in the computer program ?*") and examples in Table 2 to illustrate how SIGN identifies the semantic dissimilar and semantic similar sentence pairs, respectively. The experiments are all conducted on the QQP dataset.

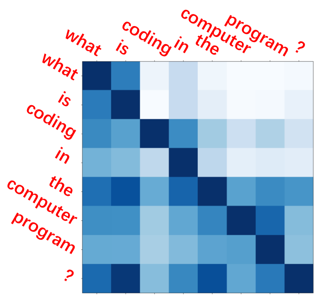
5.3.1 Semantic Dissimilar Pairs Identification. First of all, we illustrated the attribute similarities between the node in Figure 8. The similarities were calculated as the dot products of the corresponding node representations in the matching graph. From Figure 8(a), we can see that the attribute structures helped to discriminate the chunk pair "is computer programming" (whose chunk type is VP) and "in the computer program" (whose chunk type is PP), although these two chunks are similar in form (i.e., they share the words "computer" and "program" after stemming). More interestingly, the chunk "is computer programming" has some similarities to "is coding." The reason is that both of them are "VP," and they play similar syntactic roles in the sentences (the



(a) Word-chunk similarities in X of SIGN



(b) Word-word similarities in X of SIGN



(c) Word-word similarities in X of BERT$_{BASE}$

Fig. 9. Inter-sentence chunk and word similarities for a sentence pair. Elements in rows denote the in-degree similarities and elements in columns denote the out-degree similarities. Darker means a higher value. The SIGN and BERT$_{BASE}$ models are trained on the QQP dataset.

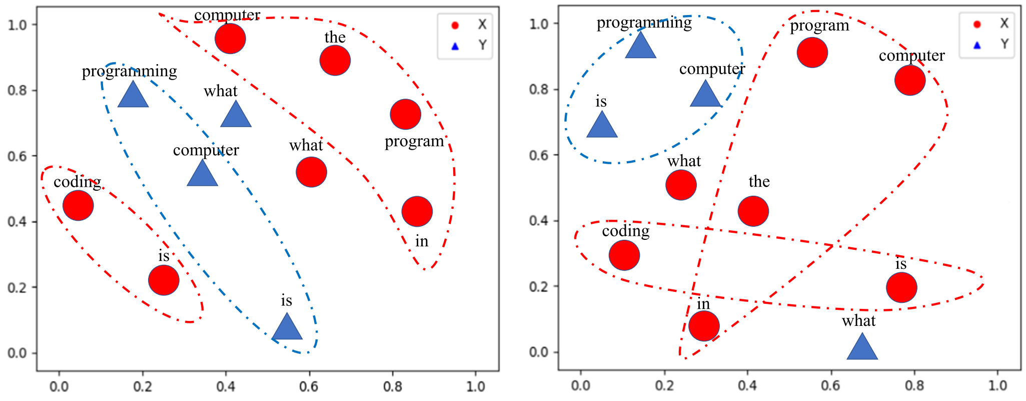
syntactic dependency trees can be found in Figure 1). Moreover, Figure 8(b) shows the cross-sentence similarities between the words. Still, we can see that the attribute structures helped SIGN eliminate most of the noisy matching signals. In contrast, the similarities by RoBERTa$_{LARGE}$ (Figure 8(c)) are dense and difficult to explain. Summarizing the results, we can conclude that the attribute structures can help SIGN filter out the noisy and mismatched signals, and highlight the true matching signals. Therefore, it enhances not only the matching accuracy but also the explainability of the model.

To further exploit the benefit of relation structure in SIGN, we conducted the word-word and word-chunk relation similarities based on the word representations by SIGN and RoBERTa$_{LARGE}$. Figure 9(a) and (b) show the similarities that were calculated as the dot products of the corresponding in-degree and out-degree node representations. From Figure 9(a), we can see that the chunk has a higher similarity with words if they have a word-chunk relation. From Figure 9(b), we can see the words have higher similarity if they have a syntactic dependency relation. Therefore, we can see that the relation structures helped SIGN capture more accurate relations in matching. In contrast, the similarities by RoBERTa$_{LARGE}$ (Figure 9(c)) are dense and noisy. The phenomenon leads to incorrect matching predictions.

To further compare the word and chunk representations by SIGN and RoBERTa$_{LARGE}$, we conducted the TSNE based on the word representations by SIGN and by RoBERTa$_{LARGE}$. Figure 10 illustrates the results. The two categories (red dots and blue triangles) respectively indicate the words from the source sentence and the target sentence. Additionally, the red dots and areas denote the words and chunks of sentence X, and the blue triangles and areas denote the words and chunks of sentence Y.

First, at a word level, we can see that the points (words) from Figure 10(a) are much easier to discriminate from than that from Figure 10(b). Second, in the chunk level, from Figure 10(a), we can see that the chunk "is computer programming" (VP) from X is closer to the chunk "is coding" (VP) from Y but has some distance to the chunk "in the computer programming" (PP). However, from the TSNE visualization of RoBERTa$_{LARGE}$ (Figure 10(b)), we can see that the chunk "is computer programming" from X is closer to the chunk "in the computer program" from Y since they are similar in form. The phenomenon leads to incorrect matching predictions.

In summary, the two figures confirm the following. First, we can get a more accurate representation of the word with its sentence context. The distance between "computer" and "programming" in **X** is further in space than "computer" and "program" in **Y** compared with SIGN and RoBERTa.



(a) Based on word representations from SIGN (b) Based on word representations from BERT$_{BASE}$

Fig. 10. TSNE illustration of a sentence pair (X = "what is coding in the computer programming?"; Y = "what is computer programming?"). The red and blue areas indicate the chunks of X and Y. The SIGN and RoBERTa$_{LARGE}$ models are trained on the QQP dataset.

Second, we can get chunking-level alignment—for example, "is coding" and "is computer programming" have closer distance compared with SIGN and RoBERTa. Third, syntactic structures help SIGN make high accurate sentence matching in an explainable way.

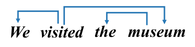
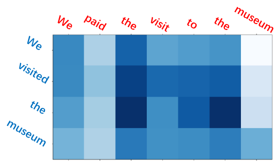
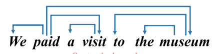
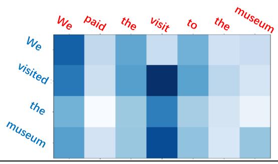
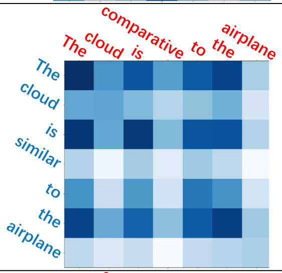
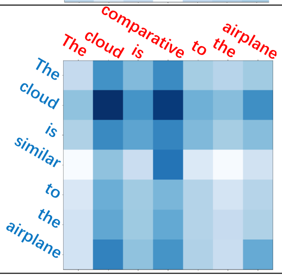
These experiments further confirmed that the relation structures help SIGN make high accurate sentence matching in an explainable way.

5.3.2 *Semantic Similar Pairs Identification.* To investigate how SIGN identified the semantic similar pairs, we conducted four case studies to respectively show the four types of syntax transformations in Table 2. The overall comparisons are shown in Tables 9 and 10. The most important syntactic structures (their β_{e_j} values are larger than 0.25) are shown in the second column. Moreover, the word-word correspondence matrices of BERT$_{BASE}$ and SIGN are shown in the fourth column. The word-word correspondences are calculated based on the word embeddings of two sentence pairs and shown as the similarity matrix. Specifically, the word-word correspondence matrix of BERT$_{BASE}$ is calculated as the dot product of two sentences' word embeddings outputted from the BERT$_{BASE}$. The word-word correspondence matrix of SIGN is calculated as the dot product of learned word embeddings $\mathbf{h}_w^L$ of two sentences (described in Section 4.5). The darker blocks mean higher similarity.

Table 9 shows the results of the syntax transformations of "different forms of a word" and "word/chunk equations or synonyms." The sentence pair (X: "we visited the museum"; Y: "We paid a visit to the museum") and pair (X: "The cloud is similar to the airplane"; Y: "The cloud is comparative to the airplane") are used as examples in the experiments. Their syntactic dependencies, chunk, and POS structures are shown in the second column. The first pair transfers the *verb* form of word ("visited") in sentence X to the word ("visit") of sentence Y to the *noun* form. This syntax transformation is referred to as verbalization. The second sentence transfers the adjective "similar" to its synonym "comparative."

From the observation of the word-word correspondence matrix of the two case studies, we can see that the matrix of BERT$_{BASE}$ focused on some function words (e.g., "(the, is)"), which are much noisier than the matrix of SIGN. SIGN, however, considers the multiple important syntactic structures and focuses on the content words (e.g., "(visit, museum, cloud, similar, comparative)"). More importantly, SIGN can identify the crucial syntactic structures and right syntax transformation.

Table 9. Case Studies for Two of Four Different Syntax Transformations Using Four Sentence Pairs in Table 2, Respectively

Syntax transforma-tion	Syntactic structures	Model	Word-word correspondence matrix
different form of a word	*X: We [visited] the museum* *Original sentence*  *Syntactic dependency* *We PR / visited VB / the DT / museum NN* *Part-of-Speech*	BERT$_{BASE}$	
	Y: We [paid a visit] to the museum *Original sentence*  *Syntactic dependency* *We PR / paid VB / a DT / visit NN / to IN / the DT / museum NN* *Part-of-Speech*	SIGN-HGAT	
word/chunk equations or synonyms	*X: The cloud is similar to the airplane* *Original sentence* *The DT / cloud NN / is VB / similar JJ / to IN / the DT / airplane NN* *Part-of-Speech* *[The cloud is similar] CLAUSE / [to the airplane] PPP* *chunk types*	BERT$_{BASE}$	
	Y: The cloud is comparative to the airplane *Original sentence* *The DT / cloud NN / is VB / comparative JJ / to IN / the DT / airplane NN* *Part-of-Speech* *[The cloud is comparative] CLAUSE / [to the airplane] PPP* *chunk types*	SIGN-HGAT	

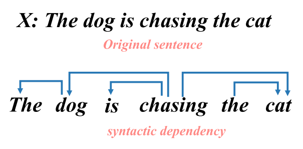
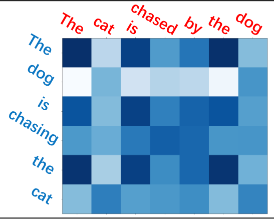
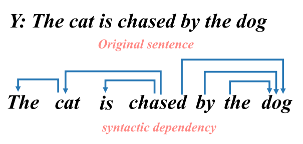
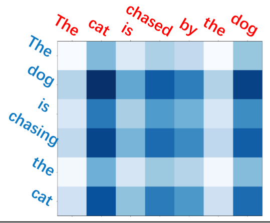
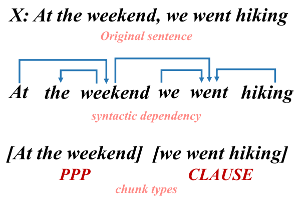
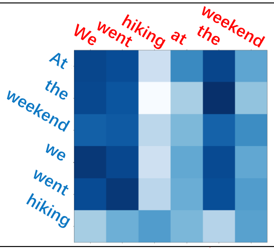
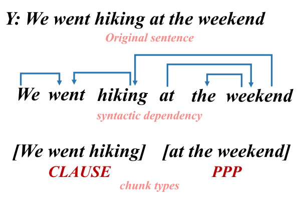
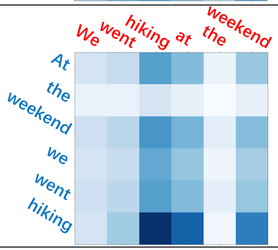
The highlighted syntactic structures of sentence pair (X, Y) are shown in the second column, and the word-word correspondence matrix of BERT$_{BASE}$ and SIGN-HGAT (the base PLM is BERT$_{BASE}$) are shown in the fourth column, respectively. Both models are trained on the QQP dataset. Darker means a higher value.

For example, the "visited-visit" similarity is darker in the first pair, and the "similar-comparative" similarity is also relatively darker.

Table 10 further shows the other two syntax transformations: "active or passive sentences" and "different word/chunk order." The sentence pair (X: "The dog is chasing the cat"; Y: "The cat is chased by the dog") and pair (X: "At the weekend, we went hiking"; Y: "We went hiking at the weekend") are used as the examples in the experiments. The first pair transfers the "active" form ("is chasing") in sentence X to the passive form ("is chased") of sentence Y. The second sentence pairs change their positions of prepositions ("at the weekend").

From the word-word correspondence matrix of the two cases, we still see that the correspondence matrix of BERT$_{BASE}$ is dense and hard to explain. However, SIGN can identify the crucial matching component through important syntactic structures. For example, the "chased-chasing"

Table 10. Case Studies for the Other Two of Four Different Syntax Transformations Utilizing Four Sentence Pairs Described in Table 2, Respectively

Syntax transformation	Syntactic structures	Model	Word-word correspondence matrix
active or passive sentence	 *X: The dog is chasing the cat* *Original sentence* *The dog is chasing the cat* *syntactic dependency*	BERT$_{BASE}$	
	 *Y: The cat is chased by the dog* *Original sentence* *The cat is chased by the dog* *syntactic dependency*	SIGN-HGAT	
different word/chunk order	 *X: At the weekend, we went hiking* *Original sentence* *At the weekend we went hiking* *syntactic dependency* *[At the weekend] [we went hiking]* *PPP* *CLAUSE* *chunk types*	BERT$_{BASE}$	
	 *Y: We went hiking at the weekend* *Original sentence* *We went hiking at the weekend* *syntactic dependency* *[We went hiking] [at the weekend]* *CLAUSE* *PPP* *chunk types*	SIGN-HGAT	

The highlighted syntactic structures of sentence pair (X, Y) are shown in the second column, and the word-word correspondence matrix of BERT$_{BASE}$ and SIGN-HGAT are shown in the fourth column, respectively. Both models are trained on the QQP dataset. Darker means a higher value.

similarity is darker for the first pair, and the "hiking-hiking" similarity is also darker for the second pair.

In summary, the example in Tables 9 and 10 verified that (1) SIGN can discriminate the important syntactic structures for matching, (2) SIGN also can identify semantic similar pairs even though their syntactic structures are different, and (3) different syntactic structures are complementary. They can help SIGN make highly accurate sentence matching in an explainable way.

6 CONCLUSION

In this article, we proposed a novel approach to learning sentence matching models based on syntactic structures, referred to as SIGN. SIGN explicitly models multiple types of syntactic structures under the framework of graph learning. The extracted syntactic structures are first used to derive a matching graph in which both the semantic and syntactic information is represented and interacted. Then, an HGAT is employed to learn the node representations for the matching graph, followed by an MLP for conducting the final matching prediction. SIGN offers several advantages: explicitly modeling multiple types and heterogeneous syntactic structures in matching, interacting with both semantic information and syntactic structures on the graph, and ease in interpretation. Experimental results based on three large-scale public benchmarks verified the effectiveness and explainability of the proposed model.

REFERENCES

[1] Steven Abney. 1996. Partial parsing via finite-state cascades. *Natural Language Engineering* 2, 4 (1996), 337–344.

[2] Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-BERT: Improving pre-trained transformers with syntax trees. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 3011–3020. https://doi.org/10.18653/v1/2021.eacl-main.262

[3] Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1957–1967. https://doi.org/10.18653/v1/D17-1209

[4] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 632–642. https://doi.org/10.18653/v1/D15-1075

[5] Fan Bu, Hang Li, and Xiaoyan Zhu. 2013. An introduction to string re-writing kernel. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. 2982–2986. http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6544

[6] Haolan Chen, Fred X. Han, Di Niu, Dong Liu, Kunfeng Lai, Chenglin Wu, and Yu Xu. 2018. MIX: Multi-channel information crossing for text matching. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18)*. ACM, New York, NY, 110–119. https://doi.org/10.1145/3219819.3219928

[7] Lu Chen, Yanbin Zhao, Boer Lyu, Lesheng Jin, Zhi Chen, Su Zhu, and Kai Yu. 2020. Neural graph matching networks for Chinese short text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 6152–6158. https://doi.org/10.18653/v1/2020.acl-main.547

[8] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and combining sequential and tree lstm for natural language inference. *ArXiv preprint abs/1609.06038* (2016). https://arxiv.org/abs/1609.06038

[9] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1657–1668. https://doi.org/10.18653/v1/P17-1152

[10] Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 468–476. https://aclanthology.org/P09-1053

[11] 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 (Volume 1: Long and Short Papers)*. 4171–4186. https://doi.org/10.18653/v1/N19-1423

[12] Tira Nur Fitria. 2021. QuillBot as an online tool: Students' alternative in paraphrasing and rewriting of English writing. *Englisia: Journal of Language, Education, and Humanities* 9, 1 (2021), 183–196.

[13] Angela D. Friederici and Jürgen Weissenborn. 2007. Mapping sentence form onto meaning: The syntax–semantic interface. *Brain Research* 1146 (2007), 50–58.

[14] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 2–13. https://doi.org/10.3115/v1/D14-1002

[15] Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *Proceedings of the 6th International Conference on Learning Representations: Conference Track Proceedings (ICLR'18)*. https://openreview.net/forum?id=r1dHXnH6-

[16] Ana C. Gouvea, Colin Phillips, Nina Kazanina, and David Poeppel. 2010. The linguistic processes underlying the P600. *Language and Cognitive Processes* 25, 2 (2010), 149–188.

[17] 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 Conference on Information and Knowledge Management (CIKM'16)*. 55–64. https://doi.org/10.1145/2983323.2983769

[18] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*, Vol. 2. 2042–2050. https://proceedings.neurips.cc/paper/2014/hash/b9d487a30398d42ecff55c228ed5652b-Abstract.html

[19] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM'13)*. ACM, New York, NY, 2333–2338. https://doi.org/10.1145/2505515.2505665

[20] Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTaiL: A textual entailment dataset from science question answering. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18), the 30th Innovative Applications of Artificial Intelligence (IAAI'18), and the 8th AAAI Symposium on Education Advances in Artificial Intelligence (EAAI'18)*. 5189–5198. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17368

[21] Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19), the 31st Innovative Applications of Artificial Intelligence Conference (IAAI'19), and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI'19)*. 6586–6593. https://doi.org/10.1609/aaai.v33i01.33016586

[22] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*. http://arxiv.org/abs/1412.6980

[23] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations: Conference Track Proceedings (ICLR'17)*. https://openreview.net/forum?id=SJU4ayYgl

[24] Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends in Information Retrieval* 7, 5 (2014), 343–469.

[25] Tao Liu, Xin Wang, Chengguo Lv, Ranran Zhen, and Guohong Fu. 2020. Sentence matching with syntax- and semantics-aware BERT. In *Proceedings of the 28th International Conference on Computational Linguistics*. 3302–3312. https://doi.org/10.18653/v1/2020.coling-main.293

[26] Xiaodong Liu, Kevin Duh, and Jianfeng Gao. 2018. Stochastic answer networks for natural language inference. *arXiv preprint abs/1804.07888* (2018). https://arxiv.org/abs/1804.07888

[27] Yang Liu, Matt Gardner, and Mirella Lapata. 2018. Structured alignment networks for matching sentences. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1554–1564. https://doi.org/10.18653/v1/D18-1184

[28] 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. *arXiv preprint abs/1907.11692* (2019). https://arxiv.org/abs/1907.11692

[29] Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Proceedings of the 26th International Conference on Neural Information Processing Systems, Vol. 1 (NIPS'13)*. 1367–1375. https://proceedings.neurips.cc/paper/2013/hash/8a0e1141fd37fa5b98d5bb769ba1a7cc-Abstract.html

[30] Nianzu Ma, Sahisnu Mazumder, Hao Wang, and Bing Liu. 2020. Entity-aware dependency-based deep graph attention network for comparative preference classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 5782–5788. https://doi.org/10.18653/v1/2020.acl-main.512

[31] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60. https://doi.org/10.3115/v1/P14-5010

[32] 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 (WWW'17)*. ACM, New York, NY, 1291–1299. https://doi.org/10.1145/3038912.3052579

[33] Al-Smadi Mohammad, Zain Jaradat, Al-Ayyoub Mahmoud, and Yaser Jararweh. 2017. Paraphrase identification and semantic text similarity analysis in Arabic news tweets using lexical, syntactic, and semantic features. *Information Processing & Management* 53, 3 (2017), 640–652.

[34] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 130–136. https://doi.org/10.18653/v1/P16-2022

[35] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 2793–2799.. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11895

[36] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2249–2255. https://doi.org/10.18653/v1/D16-1244

[37] Martin Potthast, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. 2011. Cross-language plagiarism detection. *Language Resources and Evaluation* 45 (2011), 45–62.

[38] Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. 2021. Do syntax trees help pre-trained transformers extract information? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2647–2661. https://doi.org/10.18653/v1/2021.eacl-main.228

[39] 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.

[40] Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 4411–4417. https://doi.org/10.24963/ijcai.2018/613

[41] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Co-stack residual affinity networks with multi-level attention refinement for matching text sequences. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4492–4502. https://doi.org/10.18653/v1/D18-1479

[42] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1565–1575. https://doi.org/10.18653/v1/D18-1185

[43] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Hermitian co-attention networks for text matching in asymmetrical domains. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 4425–4431. https://doi.org/10.24963/ijcai.2018/615

[44] Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. In *Proceedings of the 1st Workshop on Subword and Character Level Models in NLP*. 142–147. https://doi.org/10.18653/v1/W17-4121

[45] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations: Conference Track Proceedings (ICLR'18)*. https://openreview.net/forum?id=rJXMpikCZ

[46] Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*. 1354–1361. http://ijcai.org/Abstract/15/195

[47] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *Proceedings of the World Wide Web Conference (WWW'19)*. ACM, New York, NY, 2022–2032. https://doi.org/10.1145/3308558.3313562

[48] Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, and Michael Witbrock. 2019. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19), the 31st Innovative Applications of Artificial Intelligence Conference (IAAI'19), and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI'19)*. 7208–7215. https://doi.org/10.1609/aaai.v33i01.33017208

[49] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 4144–4150. https://doi.org/10.24963/ijcai.2017/579

[50] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 4144–4150. https://doi.org/10.24963/ijcai.2017/579

[51] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2021. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24.

[52] 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 Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 55–64. https://doi.org/10.1145/3077136.3080809

[53] Chen Xu, Jun Xu, Zhenhua Dong, and Ji-Rong Wen. 2022. Semantic sentence matching via interacting syntax graphs. In *Proceedings of the 29th International Conference on Computational Linguistics*. 938–949. https://aclanthology.org/2022.coling-1.78

[54] Jun Xu, Xiangnan He, and Hang Li. 2019. Deep learning for matching in search and recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM'19)*. ACM, New York, NY, 832–833. https://doi.org/10.1145/3289600.3291380

[55] Runqi Yang, Jianhai Zhang, Xing Gao, Feng Ji, and Haiqing Chen. 2019. Simple and effective text matching with richer alignment features. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4699–4709. https://doi.org/10.18653/v1/P19-1465

[56] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19), the 31st Innovative Applications of Artificial Intelligence Conference (IAAI'19), and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI'19).* 7370–7377. https://doi.org/10.1609/aaai.v33i01.33017370

[57] Xueli Yu, Weizhi Xu, Zeyu Cui, Shu Wu, and Liang Wang. 2021. Graph-based hierarchical relevance matching signals for ad-hoc retrieval. In *Proceedings of the Web Conference 2021.* 778–787.

[58] Bo Zhang, Yue Zhang, Rui Wang, Zhenghua Li, and Min Zhang. 2020. Syntax-aware opinion role labeling with dependency graph convolutional networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* 3249–3258. https://doi.org/10.18653/v1/2020.acl-main.297

[59] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-aware BERT for language understanding. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20), the 32nd Innovative Applications of Artificial Intelligence Conference (IAAI'20), and the 10th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI'20).* 9628–9635. https://aaai.org/ojs/index.php/AAAI/article/view/6510

Received 16 December 2021; revised 29 August 2022; accepted 5 July 2023