Semantic Sentence Matching via Interacting Syntax Graphs

Chen Xu[1], **Jun Xu**[1,2,*], **Zhenghua Dong**[3], **Ji-Rong Wen**[1,2]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods
[3]Huawei Noah's Ark Lab
{xc_chen, junxu, jrwen}@ruc.edu.cn
dongzhenhua@huawei.com

Abstract

Studies have shown that the sentence's syntactic structures are important for semantic sentence matching. A typical approach is encoding each sentence's syntactic structure into an embedding vector, which can be combined with other features to predict the final matching scores. Though successes have been observed, embedding the whole syntactic structures as one vector inevitably overlooks the fine-grained syntax matching patterns, e.g. the alignment of specific term dependencies relations in the two inputted sentences. In this paper, we formalize the task of semantic sentence matching as a problem of graph matching in which each sentence is represented as a directed graph according to its syntactic structures. The syntax matching patterns (i.e. similar syntactic structures) between two sentences, therefore, can be extracted as the sub-graph structure alignments. The proposed method, referred to as Interacted Syntax Graphs (ISG), represents two sentences' syntactic alignments as well as their semantic matching signals into one association graph. After that, the neural quadratic assignment programming (QAP) is adapted to extract syntactic matching patterns from the association graph. In this way, the syntactic structures fully interact in a fine granularity during the matching process. Experimental results on three public datasets demonstrated that ISG can outperform the state-of-the-art baselines effectively and efficiently. The empirical analysis also showed that ISG can match sentences in an interpretable way.

1 Introduction

Matching two natural language sentences has become a fundamental technique in information retrieval (IR) and natural language processing (NLP). Extensive research efforts have been devoted to the task (Li and Xu, 2014; Xu et al., 2020). Recently, researchers found that the sentences' rich syntactic information helps match. Some studies utilize the implicit syntax-encoding methods to learn the sentence embedding based on its syntactic strictures (Chen et al., 2016, 2017). Other studies directly utilize the different syntactic tags as syntactic features and fuse them with word features (Mou et al., 2016; Chen et al., 2018; Liu et al., 2020). All these approaches separately represent the syntactic information of the two input sentences as two coarse-grained embedding vectors, ignoring the fine-grained matching patterns between the syntactic structures.

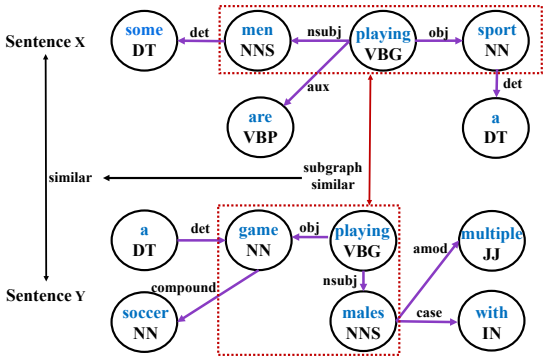


Figure 1: A semantically similar sentence pair (X = "Some men are playing a sport", Y ="A soccer game with multiple males playing") from SNLI. The words and POS tags are shown in the nodes, and syntactic dependencies are shown as edges.

In sentence matching, The fine-grained syntax structures are lost during the process of representing the syntactic information as a vector. Studies in (Xu et al., 2020) also verified that representation-based methods will lose the fine-grained matching signals. Figure 1 illustrates two semantically similar sentences from the SNLI dataset where sentences X = "Some men are playing a sport", Y ="A soccer game with multiple males playing". The parsed syntactic structures (the POS tags and syntactic dependencies) are represented as nodes and edges in two graphs, respectively. We can see that though the overall syntactic struc-

* Corresponding author

tures of the two sentences are very different, they still contains matched sub-graphs: "men(NNS) $\xleftarrow{\text{nsubj}}$playing(VBG)$\xrightarrow{\text{obj}}$sport(NN)" from X and "game(NN) $\xleftarrow{\text{obj}}$playing(VBG)$\xrightarrow{\text{nsubj}}$males(NNS)" from Y. The matched sub-graphs provide a crucial syntactic matching signal for downstream models to make high-precision matching decisions. Existing approaches, however, inevitably overlook the fine-grained sub-graph matching signal because the sub-graph details are lost during the embedding of the whole syntax graphs. Specifically, the sub-graph details denote the combination of first-order (e.g. Part-of-Speech) and second-order (e.g. word dependency) syntactic information.

In this paper, we propose to formalize the task of sentence matching as matching two directed graphs where each graph corresponds to one sentence, and its edges and nodes correspond to the word-word syntactic dependencies and the words' syntax and semantic information, respectively. The sentence matching, therefore, becomes a process of first extracting the sub-graph structure alignmentsand then summarizing them into the final matching score.

A neural model called Interacted Syntax Graphs (ISG) is developed for conducting the matching. Specifically, given two sentences, ISG employs a pre-trained language model (PLM) and a syntactic parser to get the word embeddings and the syntactic structures as the initialization features. Then, ISG fuses these semantic and syntactic features into an associate graph. Sub-graph matching patterns can be extracted based on the associate graph, by using Lawler's quadratic assignment programming (QAP) (Cho et al., 2010). Finally, a matching classifier is used to merge the matching patterns and semantic vectors outputted from the PLM, resulting in the final matching score.

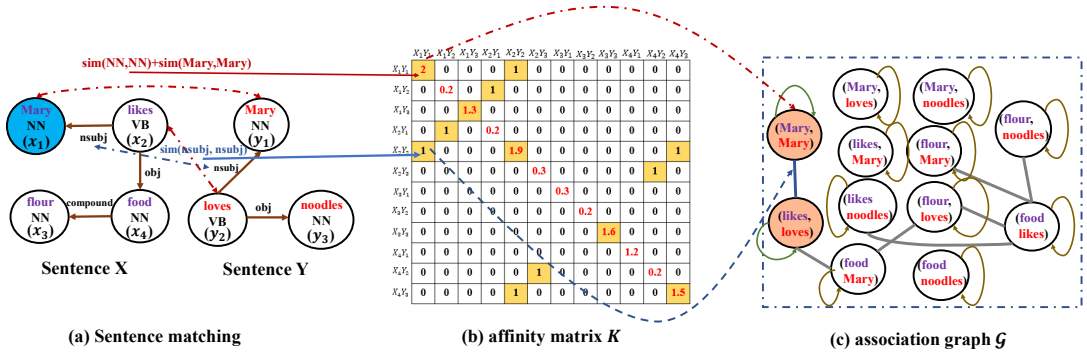
ISG offers several advantages, including accurate extraction and fusion of the syntactic graph matching signals, ease in interpretation, and high matching accuracy. The contributions of this paper can be summarized as follows: (1) We highlight the importance of the fine-grained matching patterns from syntax graphs in semantic sentence matching. A novel matching model called ISG is proposed; (2) Experimental results based on three available benchmarks showed that the matching accuracy of ISG outperformed the state-of-the-art baselines; (3) Analysis showed that ISG can discriminate important syntactic and semantic matching patterns in an interpretable way.

2 Related Work

Machine learning models have been widely used for matching natural language sentences (Li and Xu, 2014; Xu et al., 2020). Among them, the representative methods include DSSM (Huang et al., 2013) and its extensions (Wang et al., 2017a; Shen et al., 2014; Kim et al., 2019; Yang et al., 2019). Representative interaction-based models include ARC-II (Hu et al., 2014), MatchPyramid (Pang et al., 2016), etc. Recently, the pre-trained language model has been adapted to conducting matching (Devlin et al., 2019; Liu et al., 2019). These models always focused on superficial matching signals and ignore explicit NLP knowledge.

Recently, there is a trend to utilize explicit NLP knowledge to improve sentence representation. For example, HIM (Chen et al., 2017) used the syntactic dependencies to enhance the sentence representations, see also (Chen et al., 2016; Liu et al., 2018; Tymoshenko and Moschitti, 2018). The NLP knowledge-enhanced matching models have also adapted to the interaction-based models. For example, MIX (Chen et al., 2018) utilizes POS and named-entity tags as prior features. Recently, (Sachan et al., 2021; Bai et al., 2021; Zhang et al., 2020b) found that syntax can help PLM capture more information and achieve impressive results for NLP tasks. However, these models overlook the fine-grained syntactic matching patterns.

Graph matching problem (GM) has also been adopted for discovering the patterns between different graphs (Loiola et al., 2007). The key is to learn a practical affinity function with given two structures. In early work, most on seeking approximate affinity function and Euclid distance together with the Gaussian kernel is applied (Cho et al., 2010; Leordeanu et al., 2012). Recently, quadratic assignment programming (QAP) (Cho et al., 2010) has a wide application in Graph Matching (GM) because of its great performance. The affinity function in QAP can be learned with the manners of unsupervised (Leordeanu et al., 2012), semi-supervised (Leordeanu et al., 2011), or supervised (Loiola et al., 2007). Recently, deep graph matching has been applied for GM on images (Zanfir and Sminchisescu, 2018; Wang et al., 2021) and the matching accuracy has been achieved. Moreover, Graph-based models have also been used for sentence matching (Yao et al., 2019; Zhang et al., 2020a; Sachan et al., 2021).



(a) Sentence matching (b) affinity matrix K (c) association graph $\mathcal{G}$

Figure 2: Example of converting the matching of a pair of sentences with syntactic structures (a) to the corresponding affinity matrix K (b) or association graph $\mathcal{G}$ (c).

3 Problem Formulation

3.1 Sentence matching

The matching of a pair of natural language sentences can be formally described as follows: suppose that $\mathcal{Z}$ is the set of labels which is defined by a specific matching task. In the paraphrase identification (PI) tasks, $\mathcal{Z} = \{0, 1\}$, where '0' and '1' respectively denote the relationship of "dissimilar" and "similar"; in natural language inference (NLI) $\mathcal{Z} = \{0, 1, 2\}$, where 0, 1, 2 respectively indicate "contradiction", "neutral", and "entailment". A set of training instances $\mathcal{D} = \{(X_i, Y_i, z_i)\}_{i=1}^{N}$ is given where each sample $(X, Y, z) \in \mathcal{D}$ consists of a sentence pair (X, Y) and its ground-truth matching label z. Moreover, the X, Y are two sequences of words: $X = \{x_1, x_2, \cdots, x_{t_X}\}$ and $Y = \{y_1, y_2, \cdots, y_{t_Y}\}$, where the x_i and y_j denote the i-th and j-th words in X and Y, t_X and t_Y are the number of words (lengths) of X and Y, respectively.

3.2 Quadratic assignment programming

Quadratic assignment programming (QAP) is a type of combinatorial optimization problems (Loiola et al., 2007), originally designed for the facilities-location problems. Suppose to assign N facilities to N locations, with the cost f_{ij}, d_{kl} of the affinities between the facilities (i, j) and locations (k, l), plus the costs $b_{i\phi(i)}$ of assigning a facility i to a certain location $\phi(i)$. The objective of QAP is assigning each facility to a location such that the total assignment cost is minimized. Lawler (1963) introduced a general form of QAP as in Equation (1):

$$\min_{\phi \in S} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij\phi(i)\phi(j)} + \sum_{i=1}^{N} b_{i\phi(i)}, \qquad (1)$$

where S is the set of all permutations $\phi : N \to N$, $\forall i, j, k, l, c_{ijkl} := f_{ij}d_{kl}$ if $\wedge i \neq j, k \neq l$, otherwise $c_{iikk} := f_{ii}d_{kk} + b_{ik}$. The formulation has been widely applied to graph matching, which involves establishing node correspondences between two graphs based on the linear and quadratic structure affinity (Leordeanu and Hebert, 2005).

3.3 Sentence matching over syntax graphs

This paper proposes to adopt QAP for conducting sentence matching, by regarding the words in one sentence as the "facilities" and words in another sentence as the "locations", and their differences in syntactic structures and semantics as the "assignment costs". In this way, QAP enables the matching model to involve not only the linear syntactic structure (e.g. word attribute structure) costs which correspond to assigning the "facilities" to certain "locations", but also the quadratic syntactic structures (e.g. word-word relation structure) costs which correspond the affinities between the assigning "facilities" and "locations".

Figure 2 gives an illustrative example of formulating the sentence matching as graph matching and finding matching patterns in the affinity graph. with a sentence pair X ="*Mary likes flour food*" (length $|X| = 4$) and Y ="*Mary loves noodles*" (length $|Y| = 3$), using the parsed POS tags and syntactic dependencies shown in Figure 2(a) which are represented as the node and edge weights, respectively.

As shown in Figure 2(a), we firstly constructed two directed graph (G_X, G_Y) based on the two sentences (X, Y)'s syntax. Specifically, the word embedding and word syntax (i.e., POS) can be encoded as node features, and the word-word syntax (i.e., dependencies) can be encoded as edge features. The fine-grained syntax-based matching signals can be viewed as the alignments between

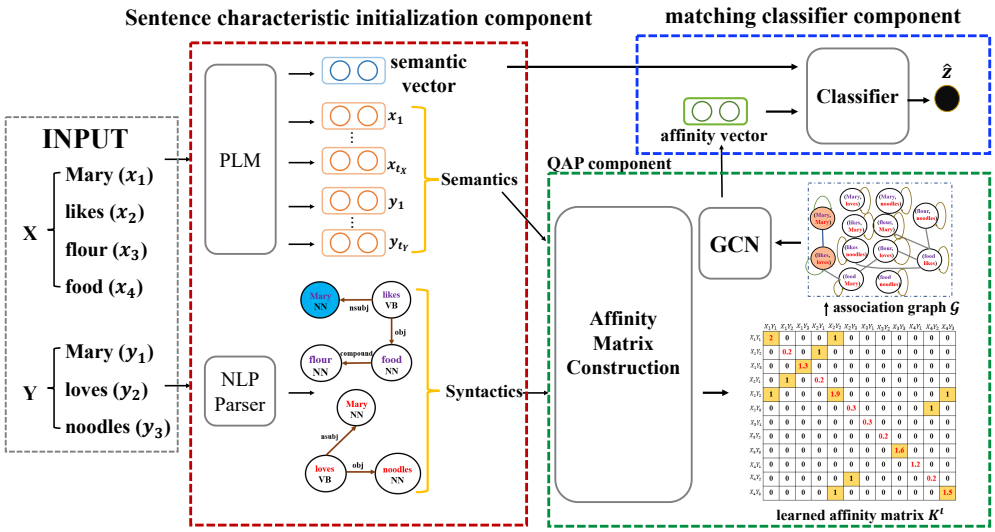


Figure 3: Architecture of Neural Quadratic Assignment Programming for Sentence Matching.

sub-graphs of two sentence graphs (G_X, G_Y).

At the same time, to capture syntax-based patterns effectively, we further construct the affinity matrix K or the association graph $\mathcal{G}$ based on the sentence graphs (G_X, G_Y). The weights of nodes and edges in the association graph are corresponding to the diagonal and off-diagonal elements in the affinity matrix (Figure 2(b,c)), respectively.

Specifically, the weights of nodes describe the word semantic similarities and POS (word attribute) affinities, and the weights of edges describe the syntactic dependency (word-word relation) affinities. For example, the node x_1y_1 =("Mary","Mary") could have the weight of, for example, $1.0 + 1.0 = 2.0$ where the first 1.0 denoting the semantic similarity, and the second 1.0 denoting the similarity between the POS tags.

As an example for the edges corresponding to the off-diagonal elements in affinity matrix K, there could be an edge with weight, for example, 1.0 between node x_1y_1 =("Mary","Mary") and node x_2y_2 =("likes","loves") because the dependency relation between x_1 ="Mary" and x_2 ="likes" is "nsubj", while the dependency relation between y_1 ="Mary" and y_2 ="loves" is also "nsubj". Similarly, the other edges can also be created.

Then the QAP is applied to learn the semantic and syntactic matching patterns in affinity matrix K. Formally, a relaxed form of QAP can be shown as Equation (2):

$$\max_{S} \ \mathrm{vec}(\mathbf{S})^T \mathbf{K} \mathrm{vec}(\mathbf{S}), \tag{2}$$

where $\mathbf{S}$ matrix encodes the word-word correspondence; $\mathrm{vec}(\mathbf{S})$ is $\mathbf{S}$'s column-vectorized notation, and $\mathbf{K} \in \mathbb{R}^{t_X t_Y \times t_X t_Y}$. The $\mathbf{S}$ matrix can be regarded as an aggregated matching patterns of syntax and semantics.

4 Proposed model: ISG

In this section, we present an efficient implementation of interacting syntax graphs (ISG) with quadratic assignment programming (QAP) (Lawler, 1963; Cho et al., 2010). Figure 3 illustrates the model architecture, which can be divided into sentence characteristic initialization, QAP component, and matching classifier.

4.1 Sentence characteristic initialization

In this component, the inputted natural language sentence pair (X, Y) is processed with a pre-trained language model (PLM) and an NLP parser, generating the semantic features and syntactic structures.

Semantic features Given a pair (X, Y), the semantic matching vector (e.g., "[CLS]" vector of BERT), $\mathbf{v}_s \in \mathbb{R}^d$ and the words embeddings $\mathbf{F}^X \in \mathbb{R}^{t_X \times d}, \mathbf{F}^Y \in \mathbb{R}^{t_Y \times d}$ consists of the semantic features:

$$(\mathbf{v}_s, \mathbf{F}^X, \mathbf{F}^Y) = \mathrm{PLM}(X, Y; \theta_p),$$

where d denotes the size of the feature vector, PLM could be BERT or other PLM models, and θ_p denotes the parameters of PLM.

Syntactic structures Generally speaking, there are two types of structures: the word attribute structure (WAS) which reflects the attributes of the word, and the word-word relation structure (WRS) which defines the relationship between two words.

The WAS attributes can be further categorized

and this paper only considers POS attributes. Given any sentence $X = \{x_1, \cdots, x_{t_X}\}$, the sequence of WAS attributes could be $\left\{a_{x_1}, a_{x_2}, \cdots, a_{x_{t_X}}\right\}$.

The WRS attributes can also be further categorized and this paper considers syntactic dependency. Given any sentence $X = \{x_1, \cdots, x_{t_X}\}$, the WRS parsing results (a dependency parsing graph) can be represented as two incidence matrices:

$$(\mathbf{I}^X, \mathbf{H}^X) = \text{Parse}(X),$$

where $\mathbf{I}^X \in \mathbb{R}^{t_X \times e_X}$ records the output-links and $\mathbf{H}^X \in \mathbb{R}^{t_X \times e_X}$ records the in-links, e_X denotes the edge number of WRS. The elements of these two matrices are defined as: if k-th edge links from word x_i to x_j (its type also denoted as $e_k(x_i, x_j)$), $\mathbf{I}^X(i,k) = \mathbf{H}^X(j,k) = 1$, and note that in order to reduce the noise from the dependencies, we also set $\mathbf{I}^X(j,k) = \mathbf{H}^X(i,k) = 1$. Otherwise, $\mathbf{I}^X(i,k) = \mathbf{H}^X(j,k) = \mathbf{I}^X(j,k) = \mathbf{H}^X(i,k) = 0$.

In this paper, we used the Stanford CoreNLP parser (Manning et al., 2014) for getting POS, and syntactic dependencies. Note that other syntactic structures can be also used, such as named-entity and semantic dependencies (Wang et al., 2019b).

4.2 QAP component

Based on the word embeddings and parsed syntactic structures, the QAP component first constructs an association graph (affinity matrix) and then solves the QAP problem, achieving the permutation which represents the word matching between the two sentences.

4.2.1 Learned affinity matrix construction

Following the practices in (Zhou and De la Torre, 2015), the QAP sparse affinity matrix $\mathbf{K}^l \in \mathbb{R}^{t_X t_Y \times t_X t_Y}$, referred to as the learned affinity matrix, can be factorized as

$$\mathbf{K}^l = \text{diag}(\text{vec}(\mathbf{P})) + (\mathbf{I}^X \otimes_{\mathcal{K}} \mathbf{I}^Y)\text{diag}(\text{vec}(\mathbf{R}))(\mathbf{H}^X \otimes_{\mathcal{K}} \mathbf{H}^Y)^T, \tag{3}$$

where operator diag$(\cdot)$ builds a diagonal matrix from input vector, $\mathbf{I}^X, \mathbf{H}^X, \mathbf{I}^Y, \mathbf{H}^Y$ are sentences X and Y's parsing results, as described in Section 4.1, $\otimes_{\mathcal{K}}$ denotes Kronecker product, and $\mathbf{P}$ and $\mathbf{R}$ encode the WAS, word embedding similarity and WRS similarity matrix, respectively and they are defined as:

$$\mathbf{P} = (1-\alpha)\mathbf{U}^X \mathbf{\Lambda_u} \mathbf{U}^{Y^T} + \alpha \mathbf{F}^X \mathbf{\Lambda_f} \mathbf{F}^{Y^T}, \mathbf{R} = \mathbf{L}^X \mathbf{\Lambda_r} \mathbf{L}^{Y^T},$$

where $\mathbf{\Lambda_u}$,$\mathbf{\Lambda_f}$,$\mathbf{\Lambda_r}$ are learn-able parameters for affinity metric, α is the trade-off coefficient for POS affinities and word-word similarities, and $\mathbf{U}^X \in \mathbb{R}^{t_X \times d}, \mathbf{U}^Y \in \mathbb{R}^{t_Y \times d}$ are the WAS sequence embeddings of X, Y and the edge representations $\mathbf{L}^X \in \mathbb{R}^{e_X \times d}, \mathbf{L}^Y \in \mathbb{R}^{e_Y \times d}$ are built by its edge sequence embeddings. Note that all the aforementioned operations for constructing $\mathbf{K}^l$ allow back propagation, and we adopt the GPU implementation provided by (Wang et al., 2019a). A more detailed QAP factorization is given in Appendix A.

4.2.2 Solving the permutation vector

Due to the high compute cost for solving the permutation vector through the learned affinity matrix K^l, we adopt the GCN method implemented by Wang et al. (2019a) to approximate the QAP problem into a linear assignment programming(LAP) problem, which can be solved in an efficient way for both time and space.

Specifically, we build the association graph $\mathcal{G} = \{\mathbf{v}^{(0)}, \mathbf{A}\}$ with its initial node embedding $\mathbf{v}^{(0)}$ and its sparse adjacent matrix $\mathbf{A}$ from the learned affinity matrix K^l. Then we can apply GCN method to updated the node embedding for k-th GCN layer, $k = 1, 2, \cdots, G_k$. The key idea is to encode the quadratic structure (WRS) to the linear structure (WAS). The permutation matrix S can be regarded as the last layer of the node features:

$$\text{vec}(S) = \mathbf{v}^{(G_k)}, \ \mathbf{v}^{(k+1)} = \mathbf{A}\mathbf{W} f(\mathbf{v}^{(k)}; \theta_k)) + \mathbf{v}^{(k)}, \tag{4}$$

where the $f(\cdot)$ is a MLP projection function at the k-th layer is parameterized by θ_k and the k-th layer node embedding of association graph denotes as: $\mathbf{v}^{(k)} \in \mathbb{R}^{t_X t_Y \times \ell^k}$, with the initial embeddings $\mathbf{v}^{(0)} \in \mathbb{R}^{t_X t_Y \times 1}$ taken from the diagonal elements of K^l. The GCN projection matrix $\mathbf{W} \in \mathbb{R}^{t_X t_Y \times t_X t_Y}$ comes from the off-diagonal elements.

$$\mathbf{v}^{(0)}(i,a) = \mathbf{K}^l(ia, ia), \ \mathbf{W}(ia, jb) = \mathbf{K}^l(ia, jb),$$

for all $i, j \in t_X, a, b \in t_Y$.

As for the adjacent matrix $\mathbf{A}$, in order to control its sparsity, we introduce a hyper-parameter γ to generate the sparse adjacent matrix of association graph $\mathcal{G}$ from the projection matrix $\mathbf{W}$:

$$A(ia, jb) = \begin{cases} 1 & \text{if } W(ia, jb) \geq \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

4.3 Matching classifier

Given the semantic matching feature $\mathbf{v}_s$ and QAP permutation vector $\mathbf{v}^{(G_k)}$, the matching score $\hat{z}$ can

be obtained by the MLP parameterized by θ_m:

$$\hat{\mathbf{z}}(X,Y) = \text{MLP}([\mathbf{v}_s|\mathbf{v}^{(G_k)}];\theta_m). \quad (5)$$

where '|' denotes the concatenation operation, $\hat{\mathbf{z}}(X,Y) = \left[\hat{z}_1, \cdots, \hat{z}_{|\mathcal{Z}|}\right]$ and $\hat{z}_k$ denotes the probability of k-th category. The last layer is softmax so that the output is a probability distribution.

4.4 Learning the model parameters

ISG has parameters to determine, including $\mathbf{\Theta} = \{\theta_p, \theta_k, \mathbf{\Lambda_u}, \mathbf{\Lambda_f}, \mathbf{\Lambda_r}, \theta_m \mid k = 1, 2, \cdots, G_k\}$. In the training phase, given a set of sentence pairs with ground truth labels $\mathcal{D} = \{(X_i, Y_i, z_i)\}_{i=1}^N$, the learning algorithm aims to minimize the matching loss $\mathcal{L}_m$ which measures the differences between the prediction $\hat{z}$ and ground-truth z, regularized by the affinity regularizer $\mathcal{R}$ which forces the learned affinity matrix K^l and the original parsed affinity matrix K the being similar. Formally, the loss $\mathcal{L}$ that being minimized is:

$$\mathcal{L} = \mathcal{L}_m(\hat{z}, z) + \lambda_a \mathcal{R}(K, K^l) + \mu_r \|\theta\|^2, \quad (6)$$

where $\|\theta\|^2$ is the ℓ_2 regularizer, λ_a, μ_r denote the trade-off coefficient of affinity regularizer and ℓ_2 regularizer.

Matching loss The matching loss $\mathcal{L}_m$ is learned by minimizing the cross-entropy loss between the labels and the predicted results:

$$\mathcal{L}_m = -\sum_{(X,Y,\mathbf{z})\in\mathcal{D}} \sum_{k=1}^{|\mathcal{Z}|} z_k \log \hat{z}_k, \quad (7)$$

Affinity regularizer The affinity regularizer $\mathcal{R}$ aims to force the structure affinities respectively correspond to the parsed syntactic structure and that of learned from neural network to be similar. Thus the $\mathcal{R}$ is learned to minimize the KL-divergence between the learned affinity matrix K^l and parsed affinity matrix K:

$$\mathcal{R} = \sum_{(X,Y)\in\mathcal{D}} KL(K||K^l), \quad (8)$$

where the parsed affinity matrix K is defined as follows: the diagonal elements $K(ia, ia)$ will be 1 if the matched words have identical attribute, otherwise 0. And the off-diagonal element $K(ia, jb)$ will be 1 if the word pair (x_i, x_j) and (y_a, y_b) have identical word-word relation, otherwise 0.

4.5 Time complexity of online matching

At the online time, ISG needs to process the sentence pairs with PLM, parse them with the NLP parser, solve the QAP and finally calculate the matching score. The online time complexity for typical PLM (Devlin et al., 2019; Liu et al., 2019) and NLP parser (Manning et al., 2014; Wang et al., 2019b) 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 dimension of each word.

At the online matching, the time complexity of the relaxed QAP is related to GCN, which is of $\mathcal{O}(G_k m\ell + G_k n\ell^2)$ (**?**) on the association graph, where $n = t_X t_Y$ is the total number of nodes, m is the total number of edges, G_k is the number of layers, and ℓ is the dimension of the node hidden features. Note that the hyper-parameter γ controls the sparsity of the edges (as mentioned in Section 4.2), we can adjust γ so that $m \ll t_X t_Y$ and therefore reduce the time complexity of the relaxed QAP to $\mathcal{O}(G_k t_X t_Y \ell^2)$, which is more efficient than the original QAP (Wang et al., 2019a). Therefore, the total time complexity of ISG is $\mathcal{O}(|t_X + t_Y|^2 \times d + G_k t_X t_Y \ell^2)$, which is comparable with the underlying PLM.

5 Experiments

We conducted experiments to verify the effectiveness of the proposed approach. The source code and experiments are available at the link.[1]

5.1 Experimental Settings

The experiments were conducted on three large scale publicly available benchmarks:

Quora Question Pairs (QQP):[2] a large public dataset for paraphrase identification. QQP contains 404k labeled sentence pairs. We used the same data split as in (Wang et al., 2017b). **SNLI:**[3] a well-known dataset for natural language inference (NLI). SNLI contains 570k labeled sentence pairs. Following the practices in (Bowman et al., 2015), we used the same data split way. **SciTail:**[4] another NLI dataset based on science exams and web. Its label only contains two classes: "entailment" or "neutral". The dataset contains 27k sentence pairs.

[1] https://github.com/XuChen0427/Semantic-Sentence-Matching-via-Interacting-Syntax-Graphs

[2] https://www.kaggle.com/c/quora-question-pairs

[3] https://nlp.stanford.edu/projects/snli

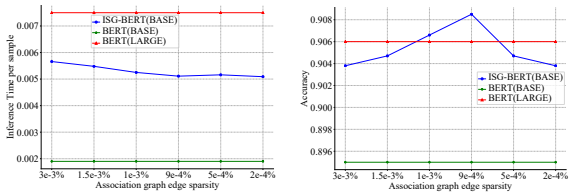
[4] http://data.allenai.org/scitail/

Several state-of-the-art baselines which conducts the matching without utilizing syntactic structures were chosen as the baselines, including DIIN (Gong et al., 2018), MwAN (Tan et al., 2018),BIMPM (Wang et al., 2017a), CSRAN (Kim et al., 2019), DecAtt (Parikh et al., 2016), CAFE (Tay et al., 2018), and DGEM (Khot et al., 2018), RE2 (Yang et al., 2019), and the BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019). Some models are task-adopted (e.g. DGEM is for NLI task), thus they are missing on some datasets. ISG was compared with the baselines DDR-match (Yu et al., 2020, 2022) that applied unsupervised assignment problems to conduct sentence matching. ISG was also compared with the baselines that utilize syntactic structures like HIM (Chen et al., 2017), which uses the constituency tree to improve local word representation, and Sembert (Zhang et al., 2020b), Syntax-bert (Bai et al., 2021) that applied different syntax to improve PLM performance. To make a fair comparison, we cannot reproduce their results due to the source codes are unavailable public of graph-based matching models (). Therefore, we decided to implement a representation-based GCN method, by following the representation-model architecture (lines 455-461, 497-505) and using the same parsing results. The method is denoted as ISG (representation-GCN) in Table 1.

To get the syntactic structures of the inputted sentences, the Stanford-corenlp (Manning et al., 2014) was used to parse the syntactic structures. In all of the experiments, the maximum sentence length was set to 70 and the sentences with lengths less than 3 were removed for reducing the noise. In the training process, all of the models were trained with the learning rate tuned amongst $[1e-5, 5e-5]$. The batch size was tuned amongst $[8, 16, 32]$, and the graph network layer G_k was tuned amongst $[1, 3]$, the coefficient $\alpha = 0.8$ and the sparsity threshold tuned amongst $[0, 0.3]$. The trade-off coefficient of affinity regularizer λ_a's were tuned amongst $[4e-3, 1e-2]$.

5.2 Experimental results

Table 1 reports the matching accuracy of the proposed ISG and the baselines on the three datasets. The '-' means the number is not available. The accuracy of baselines is according to the numbers reported. For our methods, the averaged numbers over 5 runs are reported, with the standard deviations in parentheses. From the results, we can see that different versions of the proposed ISG outperformed all of the baselines. The results also indicated that though PLM (e.g. BERT,RoBERTa) achieved SOTA accuracy, ISG can still get improvements by incorporating the syntactic information.



(a) Inference time comparsion (b) Accuracy comparison

Figure 4: ISG-BERT$_{BASE}$'s inference time (figure (a)) and matching accuracy (figure (b)) curves w.r.t. the sparsity of the association graph. Experiments were conducted on SciTail.

We also note that ISG outperformed the baselines that utilize the syntactic structures for matching. Comparing ISG with these models, we found that these baseline models all encode the syntactic structures as sentence features to enrich their representations, while ISG incorporates the syntactic and semantic matching patterns through a graph matching task and aggregates them through the affinity matrix. Moreover, we also compared the graph matching methods that separately encode two graphs into two embeddings using GCN methods and conducted matching scores based on the learned embeddings. We found that ISG will still outperform the representation-based graph matching methods. The results demonstrated that the ISG is more effective to utilize syntactic matching signals.

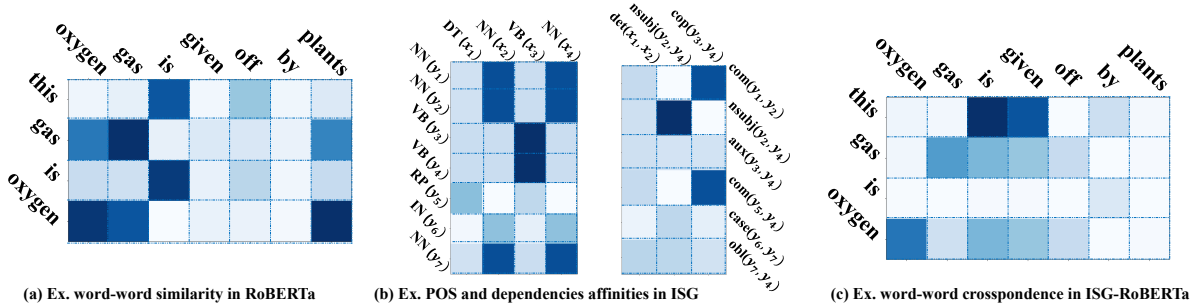
We also investigated the online time complexity of ISG. Figure 4 reports the impacts of association graph sparsity on ISG-BERT$_{BASE}$ on the Scitail test-set, where the sparsity (calculated as the fraction of edge number and square of node number in association graph) is from $[2e-4\%, 3e-3\%]$. The sparsity was adjusted by changing the hyper-parameters γ.

Figure 4(a) illustrates that the inference time of ISG will decrease with the increase of the association graph sparsity. Moreover, the inference time of ISG-BERT$_{BASE}$ is about 2.5 times that of the underlying PLM, and about 0.7 times that of BERT$_{LARGE}$. The results verified the time complexity analysis conclusion in Section 4.5.

Figure 4(b) shows the accuracy curves of ISG,

Table 1: Performance comparisons on Quora Question Pairs, SNLI and SciTail. The ±numbers in brackets mean 1-std deviations. The $*$ denotes the models are our implementations and are trained among same settings.

Models without syntactic structures	QQP:Acc(%)	SNLI:Acc(%)	SciTail:Acc(%)
DGEM (Khot et al., 2018)	-	-	77.3
DecAtt (Parikh et al., 2016)	-	82.5	81.7
CAFE (Tay et al., 2018)	-	88.5	83.3
BIMPM (Wang et al., 2017a)	88.7	88.8	85.4
DIIN (Gong et al., 2018)	89.1	-	-
MwAN (Tan et al., 2018)	89.1	-	-
CSRAN (Kim et al., 2019)	89.2	88.7	86.7
RE2 (Yang et al., 2019)	89.2	89.0	86.6
DDR-Match(BERT)* (Yu et al., 2022)	89.6	89.2	90.3
BERT$^*_{BASE}$ (Devlin et al., 2019)	89.4	89.0	89.5
BERT$^*_{LARGE}$ (Devlin et al., 2019)	89.6	89.2	90.6
RoBERTa$^*_{LARGE}$ (Liu et al., 2019)	90.0	90.1	91.5
Models with syntactic structures	QQP:Acc(%)	SNLI:Acc(%)	SciTail:Acc(%)
HIM (Chen et al., 2017)	88.7	88.6	71.6
SemBERT$^*_{BASE}$ (Zhang et al., 2020b)	89.8	90.2	92.1
SemBERT$^*_{LARGE}$ (Zhang et al., 2020b)	90.7	91.0	92.1
SyntaxBERT$_{BASE}$ (Bai et al., 2021)	89.6	87.8	-
SyntaxBERT$_{LARGE}$ (Bai et al., 2021)	89.5	89.0	-
ISG(representation-GCN)-BERT$^*_{BASE}$	90.1	89.7	90.6
ISG(representation-GCN)-BERT$^*_{LARGE}$	90.2	89.9	91.7
ISG(representation-GCN)-RoBERTa$^*_{LARGE}$	90.8	90.4	92.3
Ours(ISG-BERT$^*_{BASE}$)	**90.5** (±0.14)	**90.0** (±0.16)	**90.8** (±0.26)
Ours(ISG-BERT$^*_{LARGE}$)	**90.8** (±0.08)	**90.4** (±0.03)	**91.9** (±0.24)
Ours(ISG-RoBERTa$^*_{LARGE}$)	**91.4** (±0.1)	**91.2** (±0.08)	**93.3** (±0.2)



(a) Ex. word-word similarity in RoBERTa (b) Ex. POS and dependencies affinities in ISG (c) Ex. word-word crosspondence in ISG-RoBERTa

Figure 5: Cross sentence word-word similarity matrix and syntactic affinity matrices for two pairs : example("this gas is oxygen","oxygen gas is given off by plants"), which is from Scitail training set. Darker colors means higher similarities or affinities values.

which first increases in $[9e-4\%, 3e-3\%]$ and then dropped. We conclude that even association graph became sparse, ISG still constantly outperformed BERT$_{BASE}$ and outperformed BERT$_{LARGE}$ at some point. The results demonstrated that the QAP is efficient and will not delay the online time.

5.3 Empirical Analysis

5.3.1 Ablation Study

Firstly, we respectively set the WAS(POS) features $\mathbf{U}^X, \mathbf{U}^Y$, WRS(syntactic dependencies) features $\mathbf{L}^X, \mathbf{L}^Y$ and semantic features $\mathbf{v}_s, \mathbf{F}^X, \mathbf{F}^Y$ to zero vectors, to investigate their effects. Table 2 reports the accuracy of the ISG variation on the SciTail test data under BERT$_{BASE}$, where each variation is denoted as, for example, "ISG-w/o WRS" which means the WRS features were set zeros. Similar phenomenons have also been observed on the other two datasets, with other PLMs.

Compared ISG-BERT$_{BASE}$ with its variations, we can see that the matching performances dropped with large margins if the semantic features were set as zeros, indicating that only considering the syntax patterns did not work well. We also ob-

Table 2: Ablation study on SciTail test set.

Ablation Study Model	Acc(%)
BERT$_{BASE}$ (Devlin et al., 2019)	89.5 ($\pm$0.28)
ISG-w/o semantic and WRS	68.1 ($\pm$0.29)
ISG-w/o semantic and WAS	67.8 ($\pm$0.28)
ISG-w/o semantic	68.5 ($\pm$0.26)
ISG-w/o WAS	90.2 ($\pm$0.27)
ISG-w/o WRS	90.4 ($\pm$0.28)
ISG	**90.8** ($\pm$0.26)

Table 3: Ablation study for different syntactic structure on SciTail test set.

Ablation Study Model	Acc(%)
BERT$_{BASE}$ (Devlin et al., 2019)	89.5 ($\pm$0.28)
ISG NER&Syntactic dependencies	90.4 ($\pm$0.25)
ISG NER&Semantic dependencies	90.2 ($\pm$0.17)
ISG POS&Syntactic dependencies	**90.8** ($\pm$0.26)
ISG POS&Semantic dependencies	90.6 ($\pm$0.21)

served that the performances dropped when the WAS and WRS features were set to zeros. The bad performances were caused by removing the WAS and WRS features, indicating that syntax matching patterns are effective for sentence matching.

Moreover, We conduct the experiments with different WAS and WRS. Specifically, we respectively utilize the POS and named-entity(NE) as WAS and respectively utilize the syntactic dependencies and semantic dependencies as WRS. For semantic dependencies parsing, we follows Wang et al. (2019b). Table 3 also reports the accuracy of the ISG-BERT$_{BASE}$ variation on the SciTail test data.

Compared to the ISG variations, we can see that the matching accuracy is different for different WAS and WRS. The best and worst performance are caused by POS&syntactic dependencies and NER&Semantic dependencies, respectively. However, we can observe that all of these variations outperform the BERT baseline, which indicates the effectiveness of ISG in different WAS and WRS.

An experiment on the robustness of ISG's parameters can be found in Appendix B.

5.3.2 Matching Visualization of ISG

We conducted experiments to investigate how the ISG matched two sentences, using a real example from Scitail. The experiment was conducted based on the results of ISG-RoBERTa.

Figure 5(a) illustrated the word-word similarity matrix of these two sentences, based on the word embeddings outputted by RoBERTa, where the darker colors denote the higher similarities. Figure 5(b) illustrated the affinities between POS and dependencies in two sentences. Based on the similarities and affinity matrices, ISG solved the QAP and achieved a new correspondence matrix in Figure 5(c). The POS, word semantic similarities, and dependencies affinities correspond to the node weights and edge weights in the association graph.

The example illustrates the example that is from the Sctail training set: (X = *"this gas is oxygen"*, Y = *"oxygen gas is given off by plants"*) whose ground truth label is "neutral". The example illustrates how the ISG conducts sentence matching. Comparing word-word similarities by RoBERTa (Figure 5(a)) and that of ISG (Figure 5(c)), we can see that RoBERTa's results show high similarities between words in two sentences. On other hand, ISG can find fine-grained syntactic patterns, for example, the matching patterns of POS "VB" and "NN" and dependency "nsubj". And ISG will output the "dissimilar" result due to its low syntactic matching pattern scores.

The analysis clearly showed that ISG can well learn different syntax-based matching signals, and make them into good interactions. The results also showed what syntactic matching patterns are important and how two sentences were matched with the association graph.

6 Conclusion

This presents a novel sentence matching model which formulates sentence matching as a problem of syntax graph matching, referred to as ISG. Based on the constructed association graph, ISG explicitly aligns the syntactic sub-graphs as fine-grained matching signals. Neural QAP is adopted to learn the rich matching patterns from the training data. ISG offers several advantages: explicitly interacting with the syntactic structures in fine granularity, high matching accuracy, and the ability to interpret. Experiments on three publicly available benchmarks verified the effectiveness, robustness, and interpretability of ISG.

Acknowledgments

This work was funded by the National Key R&D Program of China (2019YFE0198200), National Natural Science Foundation of China (61872338, 61832017), and Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098.

References

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. *arXiv preprint arXiv:2103.04350*.

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 EMNLP*, pages 632–642.

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 International Conference on KDD*, pages 110–119.

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*, pages 1657–1668.

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 arXiv:1609.06038*.

Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. 2010. Reweighted random walks for graph matching. In *European conference on Computer vision*, pages 492–505. Springer.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *6th International Conference on Learning Representations, ICLR, Conference Track Proceedings*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27*, pages 2042–2050.

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 *22nd ACM International Conference on Information and Knowledge Management, CIKM'13*, pages 2333–2338.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5189–5197.

Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 6586–6593.

Eugene L Lawler. 1963. The quadratic assignment problem. *Management science*, pages 586–599.

Marius Leordeanu and Martial Hebert. 2005. A spectral technique for correspondence problems using pairwise constraints.

Marius Leordeanu, Rahul Sukthankar, and Martial Hebert. 2012. Unsupervised learning for graph matching. *International journal of computer vision*, pages 28–45.

Marius Leordeanu, Andrei Zanfir, and Cristian Sminchisescu. 2011. Semi-supervised learning and optimization for hypergraph matching. In *2011 International Conference on Computer Vision*, pages 2274–2281.

Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends in Information Retrieval*, pages 343–469.

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*, pages 3302–3312.

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*, pages 1554–1564.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. 2007. A survey for the quadratic assignment problem. *European journal of operational research*, pages 657–690.

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*, pages 55–60.

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*, pages 130–136.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2793–2799.

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 EMNLP*, pages 2249–2255.

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*, pages 2647–2661.

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 WWW*, pages 373–374.

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*, pages 4411–4417.

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*, pages 1565–1575.

Kateryna Tymoshenko and Alessandro Moschitti. 2018. Cross-pair text representations for answer sentence selection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2162–2173, Brussels, Belgium. Association for Computational Linguistics.

Runzhong Wang, Junchi Yan, and Xiaokang Yang. 2019a. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3056–3065.

Runzhong Wang, Junchi Yan, and Xiaokang Yang. 2021. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019b. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the ACL*, pages 4609–4618.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017a. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4144–4150.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, pages 4144–4150.

Jun Xu, Xiangnan He, and Hang Li. 2020. Deep learning for matching in search and recommendation. *Foundations and Trends in Information Retrieval*, pages 102–288.

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*, pages 4699–4709.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7370–7377.

Weijie Yu, Chen Xu, Jun Xu, Liang Pang, Xiaopeng Gao, Xiaozhao Wang, and Ji-Rong Wen. 2020. Wasserstein distance regularized sequence representation for text matching in asymmetrical domains. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2985–2994.

Weijie Yu, Chen Xu, Jun Xu, Liang Pang, and Ji-Rong Wen. 2022. Distribution distance regularized sequence representation for text matching in asymmetrical domains. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 30:721–733.

Andrei Zanfir and Cristian Sminchisescu. 2018. Deep learning of graph matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2684–2693.

Bo Zhang, Yue Zhang, Rui Wang, Zhenghua Li, and Min Zhang. 2020a. Syntax-aware opinion role labeling with dependency graph convolutional networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3249–3258.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020b. Semantics-aware bert for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9628–9635.

Feng Zhou and Fernando De la Torre. 2015. Factorized graph matching. *IEEE transactions on pattern analysis and machine intelligence*, pages 1774–1789.

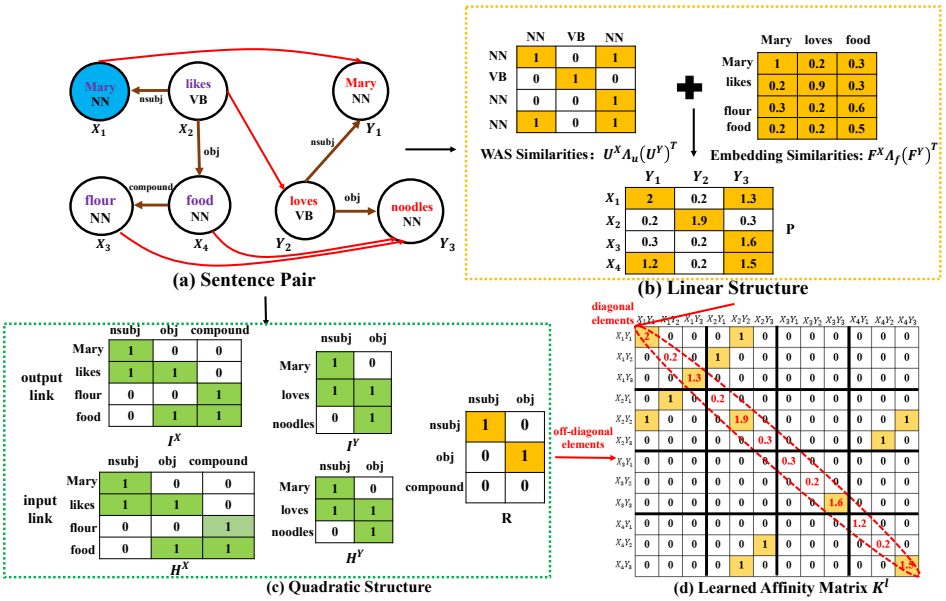


Figure 6: A working example of factorized affinity matrix K^l with aforementioned example ("Mary likes flour food", "Mary loves noodles"). The affinity matrix can be factorized into six matrices: $\mathbf{P}, \mathbf{R}, \mathbf{I}^X, \mathbf{I}^Y, \mathbf{H}^X, \mathbf{H}^Y$

Appendix A: An intuitive example on affinity matrix factorization

Figure 6 gives a working example of factorizing the affinity matrix $\mathbf{K}^l \in \mathbb{R}^{t_X t_Y \times t_X t_Y}$ in Equation (3) (Zhou and De la Torre, 2015):

$$\mathbf{K}^l = \text{diag}(\text{vec}(\mathbf{P})) + (\mathbf{I}^X \otimes_{\mathcal{K}} \mathbf{I}^Y)\text{diag}(\text{vec}(\mathbf{R}))(\mathbf{H}^X \otimes_{\mathcal{K}} \mathbf{H}^Y)^T, \quad (9)$$

with the aforementioned example sentence pair: ("Mary likes flour food", "Mary love noodles"). As shown in Figure 6(a), the words, POS and syntactic dependencies are represented in the nodes and edges, respectively.

The diagonal and off-diagonal elements in the affinity matrix (Figure 6(d)) represent the affinity of sentence linear structures and quadratic structures, respectively. According to Zhou and De la Torre (2015), the affinity matrix $\mathbf{K}^l$ can be factorized into six matrices $\mathbf{P}, \mathbf{R}, \mathbf{I}^X, \mathbf{I}^Y, \mathbf{H}^X, \mathbf{H}^Y$ (shown in Figure 6(b,c)) and defined in Section 4.

Appendix B: Robustness of ISG

ISG has a set of important hyper-parameters λ_a which trade-off the affinity regularizer $\mathcal{R}_a$ and matching loss $\mathcal{L}_m$. We conducted experiments on the Scitail test set with BERT$_{BASE}$ as the encoder to test the sensitivity of these hyper-parameters. Figure 7 illustrates the performance changes w.r.t. λ_a in terms of accuracy, where $\lambda_a \in [3e-3, 1.1e-2]$. We can see that ISG performed best when $\lambda_a \approx 8e-3$. However, the performance changes

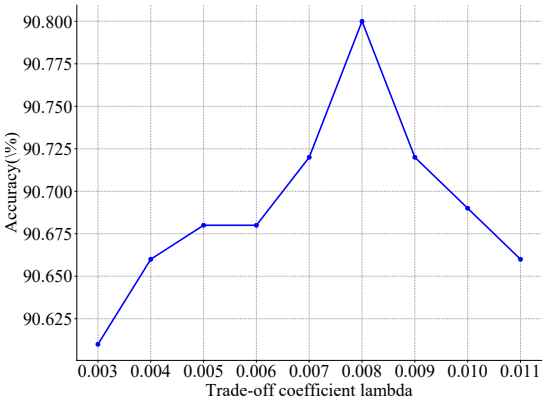


Figure 7: Accuracy curve of ISG-BERT$_{BASE}$ w.r.t. λ_a (trade-off coefficient for affinity regularizer) on the Scitail test set.

were not severe (from 90.6% to 90.8% in terms of accuracy). We conclude that (1) the introduction of the affinity regularizer enables ISG to have some tolerances to the errors caused by the NLP parser, which inevitably occurs in real-world applications; (2) ISG is robust and not sensitive to the λ_a.