Neural Quadratic Assignment Programming for Sentence Matching

Anonymous ACL submission

Abstract

Studies have shown that both the syntactic structures and words' semantics are important for sentence matching. Existing studies usually model the syntactic structures and word semantics separately, resulting in matching models that overlook the relations and dependencies between syntactic structures and semantic meanings. How to jointly model the syntactic and semantic information has become a challenging problem in sentence matching. To address the issue, we formalize sentence matching as a problem of assigning the word of one sentence to that of another sentence, with the costs determined by the differences between the corresponding syntactic structures and word embedding similarities. The proposed method, referred to as neural quadratic assignment programming for sentence matching (NQAP-SM), represents the syntactic structures and semantic matching signals as an association graph. Solving the relaxed quadratic assignment programming (QAP) on this association graph achieves the final matching score. Experimental results on three public datasets demonstrated that NQAP-SM can outperform the state-of-the-art baselines in an effective and efficient way. The analysis also showed that NQAP-SM can match sentences in an interpretable way.

1 Introduction

Matching two natural sentences has become a fundamental technique in information retrieval (IR) and natural language processing (NLP). Typical tasks include relevance ranking, paraphrase identification (PI), and natural language inference (NLI), etc. Extensive research efforts have been devoted to the task (Li and Xu, 2014; Xu et al., 2020). Most studies focus on the semantic similarities of the words/phrases of the two sentences. A number of sentence matching models have been developed, including the representation-based methods (Huang et al., 2013; Shen et al., 2014; Gao et al., 2014), interaction-based methods (Hu et al., 2014; Pang et al., 2016; Xiong et al., 2017) and their combinations (Mitra et al., 2017).

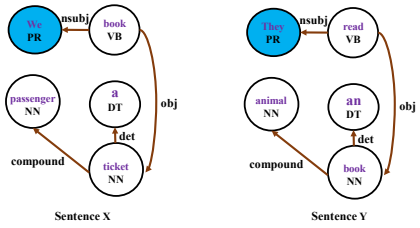


Figure 1: A pair (X = "We book a passenger ticket", Y ="they read an animal book"). The words and POS tags are shown in the nodes, and syntactic dependencies are shown as edges.

Recently, there are studies that utilize the sentences' syntactic structures for matching (Chen et al., 2016, 2017; Liu et al., 2018). Usually, the syntactic structures are firstly encoded as syntactic features and then are concatenated (or summed) with semantic features (Mou et al., 2016; Chen et al., 2018). All these methods were developed with an assumption that "two sentences with similar syntactic structures tend to be semantically similar". Syntactic features are separately extracted and used as auxiliary information to the semantic features during the matching. In real practices, however, dependencies between syntactic and semantic information can often be observed in sentence matching, which makes the assumption does not always hold.

One example is that if two sentences are very different in terms of the word semantics, they should not be considered as similar even they have identical syntactic structures. Figure 1 gives an illustrative example of two sentences "We book a passenger ticket" and "they read an animal book" with the ground-truth label "dissimilar". The parsed syntactic structures, including the POS tags and syntactic dependencies, are represented as nodes and edges in the two graphs, respectively. We can see that their POS tags and syntactic dependency structures are identical: sharing the same POS se-

quence "PR, VB, DT, NN, NN" and identical dependencies "nsubj, obj, det, compound". However, it is obvious that they are dissimilar sentences due to their totally different word semantics. We conclude that jointly modeling the words' semantic information and sentences' syntactic structures is vital for enhancing matching accuracy.

To address the issue, we propose that the matching of two sentences can be formalized as a problem of assigning the words of one sentence to the words of the other sentence, where the cost of each assignment is determined by the difference between the corresponding syntactic structures and the word embedding similarities. Intuitively, the more similar the syntactic structures and word semantics embedding of two sentences, the less the assignment cost will be spent when matching. The matching score, therefore, can be considered as the minimal cost of the whole assignment.

Specifically, we design a neural model, referred to as neural quadratic assignment programming for sentence matching (NQAP-SM), to solve the sentence QAP problem efficiently. At the online matching process, firstly, the pre-trained language model (PLM) and syntactic parser get the word embedding and syntactic structure as initialization features. Then we fuse these semantic and syntactic features into an associate graph and solve the relaxed Lawler's quadratic assignment programming (QAP) (Cho et al., 2010) on the association graph to obtain the minimal assignment results. Finally, the matching classifier merges the assignment results and semantic vector, resulting in the final matching score. During the training phase, a regularized matching loss is constructed and optimized.

NQAP-SM formulates the problem of sentence matching as a word assignment problem, which offers several advantages, including better modeling the dependency of semantic and syntactic matching signals, ease in interpretation, and improving matching accuracy. The contributions of this paper can be summarized as follows:

- We highlight the dependency between semantic and syntactic information in sentence matching. A novel matching model called NQAP-SM is proposed in which the relaxed QAP is utilized to conduct the sentence matching in an accurate, efficient, robust, and interpretable way.

- Experimental results based on three publicly available benchmarks showed that the matching accuracy of NQAP-SM outperformed the state-of-the-art baselines.

- Analysis showed that NQAP-SM not only can discriminate the similar semantic forms between sentences but also bridge the syntactic gaps between two sentences.

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; Gao 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. Mitra et al. (2017) combined both two kinds of models and improved matching accuracy. 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 the superficial matching signals and ignore the rich NLP knowledge.

Recently, there is a trend to utilize rich NLP knowledge to improve sentence matching. For example, TBCNN (Mou et al., 2016) and HIM (Chen et al., 2017) both utilize the syntactic dependency information to enhance the sentence semantic representations, see also (Chen et al., 2016; Liu et al., 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 matching signals. However, these models often separately encode the syntactic information as external features, overlooking the relation between semantic and syntactic information.

Transportation problem has also been adopted in sentence matching (Guo et al., 2016). In a transportation problem, quadratic assignment programming (QAP) (Cho et al., 2010) has a wide application in Graph Matching (GM). 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.

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 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 Formulating sentence matching with QAP

Quadratic assignment programming (QAP) is a type of combinatorial optimization problems (Loiola et al., 2007), originally designed for the facilities-location problems.

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 the 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".

When applying QAP, two main characteristics of sentence matching should be considered: the word numbers of two sentences are often different and one word from one sentence could align with multiple words from another sentence. Therefore, we further relax the one-to-one constraint condition in Lawler's QAP (Lawler, 1963)[1], which considers a relaxed form of QAP:

$$\max_{S} \ \mathrm{vec}(\mathbf{S})^T \mathbf{K} \mathrm{vec}(\mathbf{S}), \tag{1}$$

[1]In some literature, QAP also includes other special forms (Koopmans and Beckmann, 1957). .

matrix which encodes the word-word correspondence; $\mathrm{vec}(\mathbf{S})$ is $\mathbf{S}$'s column-vectorized notation, and $\mathbf{K} \in \mathbb{R}^{t_X t_Y \times t_X t_Y}$ denotes the syntactic affinity matrix whose diagonal elements encode the word-word embedding similarities and the POS affinities and its off-diagonal elements encode the syntax affinities. The perfect matching is assumed to correspond to the highest affinity score. At the same time, the affinity matrix can be converted to the association graph, whose node and edge weights can be regarded as the diagonal and off-diagonal elements of the affinity matrix, respectively.

An illustrative example of creating an association graph for a sentence pair is given in Appendix A.

In this way, sentence matching can be formulated as the Lawler's QAP (as shown in Equation (1)) through the word semantics flow over the association graph. The sentence matching score, therefore, can be viewed as the highest affinity score when word semantics flow optimally through the association graph.

4 Proposed model: NQAP-SM

In this section, we present an efficient implementation of sentence matching with QAP, called neural quadratic assignment programming for sentence matching (NQAP-SM). Figure 2 illustrates the model architecture of NQAP-SM and it can be divided into sentence characteristic initialization, QAP component and matching classifier. The next sections will describe components in details.

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

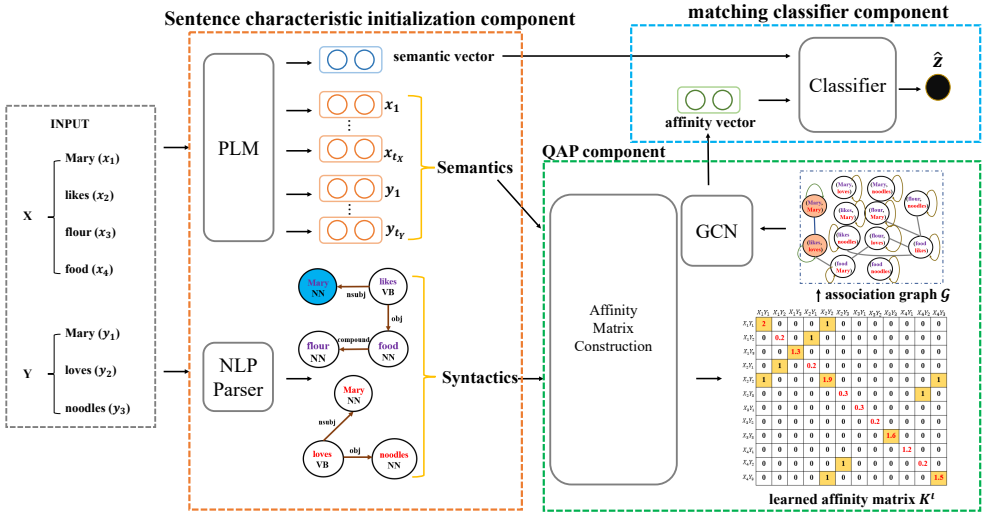


Figure 2: Architecture of Neural Quadratic Assignment Programming for Sentence Matching.

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., 2019c).

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

$$\mathbf{K}^l = \text{diag}(\text{vec}(\mathbf{P})) + (\mathbf{I}^X \otimes_{\mathcal{K}} \mathbf{I}^Y)\text{diag}(\text{vec}(\mathbf{R}))(\mathbf{H}^X \otimes_{\mathcal{K}} \mathbf{H}^Y)^T, \quad (2)$$

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).

Appendix B explanations Equation (2) with an intuitive example.

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:

$$\text{vec}(S) = \mathbf{v}^{(G_k)},\ \mathbf{v}^{(k+1)} = \mathbf{A}\mathbf{W}f(\mathbf{v}^{(k)}; \theta_k)) + \mathbf{v}^{(k)}, \tag{3}$$

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 final matching score $\hat{z}$ can be obtained by the MLP parameterized by θ_m:

$$\hat{\mathbf{z}}(X, Y) = \text{MLP}([\mathbf{v}_s | \mathbf{v}^{(G_k)}]; \theta_m). \tag{4}$$

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

NQAP-SM 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:

$$\mathcal{L} = \mathcal{L}_m(\hat{z}, z) + \lambda_a \mathcal{R}(K, K^l) + \mu_r \|\theta\|^2, \tag{5}$$

where $\|\theta\|^2$ is the ℓ_2 regularizer, λ_a^n, 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:

$$\mathcal{L}_m = -\sum_{(X,Y,\mathbf{z})\in\mathcal{D}} \sum_{k=1}^{|\mathcal{Z}|} z_k \log \hat{z}_k, \tag{6}$$

Affinity regularizer The affinity regularizer $\mathcal{R}_a$ 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 $\mathcal{L}_a$ is learned to minimize the KL-divergence between the learned affinity matrix K^l and parsed affinity matrix K:

$$\mathcal{R}_a = \sum_{(X,Y)\in\mathcal{D}} KL(K || K^l), \tag{7}$$

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 word 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, NQAP-SM needs to process the sentence pairs with PLM, parse them with 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., 2019c) 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

$\mathcal{O}(G_k m\ell + G_k n\ell^2)$ (Wu et al., 2020) 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 that the original QAP (Wang et al., 2019a). Therefore, the total time complexity of NQAP-SM 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 all of the experiments have been shared at http://github.com/hide_for_anonymous_review

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.

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. NQAP-SM was also compared with the baselines

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

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

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

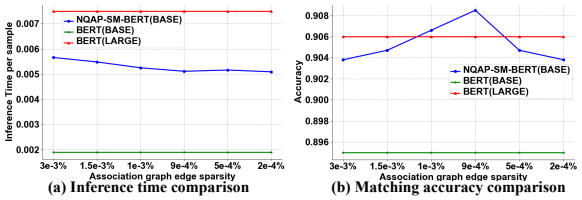


Figure 3: NQAP-SM-BERT$_{BASE}$'s inference time (figure (a)) and matching accuracy (figure (b)) curves w.r.t. the sparsity of the association graph. Experiments were conducted on SciTail.

that utilize syntactic structures like HIM (Chen et al., 2017), which uses the constituency tree)to improve local word representation. TBCNN (Mou et al., 2016) and ConSeqNet (Wang et al., 2019b) also adopt syntactic structures to the NLI tasks.

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 NQAP-SM 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 NQAP-SM outperformed all of the baselines. The results also indicated that though PLM (e.g. BERT,RoBERTa) achieved SOTA matching accuracy, NQAP-SM can still get improvements by incorporating the syntactic information.

We also note that NQAP-SM outperformed the baselines that utilize the syntactic structures for matching, with a large margin. Comparing NQAP-SM with these models, we found that these baseline models all encode the syntactic structures as sentence features to enrich its representations, while NQAP-SM incorporates the syntactic and seman-

Table 1: Performance comparisons on Quora Question Pairs, SNLI and SciTail. The ±numbers in brackets mean 1-std deviations.

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
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(%)
TBCNN (Mou et al., 2016)	-	83.5	-
ConSeqNet (Wang et al., 2019b)	-	-	85.2
HIM (Chen et al., 2017)	88.7	88.6	71.6
Ours(NQAP-SM-BERT$_{BASE}$)	**90.5** (±0.14)	**90.0** (±0.16)	**90.8** (±0.26)
Ours(NQAP-SM-BERT$_{LARGE}$)	**90.8** (±0.08)	**90.2** (±0.03)	**91.9** (±0.24)
Ours(NQAP-SM-RoBERTa$_{LARGE}$)	**91.2** (±0.1)	**90.9** (±0.08)	**93.3** (±0.2)

tic information through a relaxed QAP. The results clearly demonstrated that the QAP is more effective to utilize syntactic and semantic matching signals.

We also investigated the online time complexity of NQAP-SM. Figure 3 reports the impacts of association graph sparsity on NQAP-SM-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 through changing the hyper-parameters γ.

Figure 3(a) illustrates that the inference time of NQAP-SM will decrease with the increasing of the association graph sparsity. Moreover, the inference time of NQAP-SM-BERT$_{BASE}$ is about 2.5 times to that of the underlying PLM, and about 0.7 times to that of BERT$_{LARGE}$. The results verified the time complexity analysis conclusion in Section 4.5.

Figure 3(b) shows the accuracy curves of NQAP-SM, which first increases in $[9e-4\%, 3e-3\%]$ and then dropped. We conclude that even association graph became sparse, NQAP-SM still constantly outperformed BERT$_{BASE}$ and outperformed BERT$_{LARGE}$ at some point. The results clearly demonstrated that the QAP is efficient and will not delay the online matching time.

5.3 Empirical Analysis

We conducted experiments to analyze NQAP-SM.

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

Table 2: Ablation study on SciTail test set.

Ablation Study Model	Acc(%)
BERT$_{BASE}$ (Devlin et al., 2019)	89.5 (±0.28)
NQAP-SM-w/o semantic and WRS	68.1 (±0.29)
NQAP-SM-w/o semantic and WAS	67.8 (±0.28)
NQAP-SM-w/o semantic	68.5 (±0.26)
NQAP-SM-w/o WAS	90.2 (±0.27)
NQAP-SM-w/o WRS	90.4 (±0.28)
NQAP-SM	**90.8** (±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 (±0.28)
NQAP-SM NER&Syntactic dependencies	90.4 (±0.25)
NQAP-SM NER&Semantic dependencies	90.2 (±0.17)
NQAP-SM POS&Syntactic dependencies	**90.8** (±0.26)
NQAP-SM POS&Semantic dependencies	90.6 (±0.21)

the accuracy of the NQAP-SM variation on the SciTail test data under BERT$_{BASE}$, where each variation is denoted as, for example, "NQAP-SM-w/o WRS" which means the WRS features were set zeros. Similar phenomenons have also been observed on the other two datasets of QQP and SNLI, with other PLMs.

Compared NQAP-SM-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 syntactic structures did not work well. We also observed
when the WAS and WRS features were set to zeros. The bad performances were caused by removing the WAS and WRS features, indicating that using different types of syntactic structures is reasonable and effective for sentence matching.

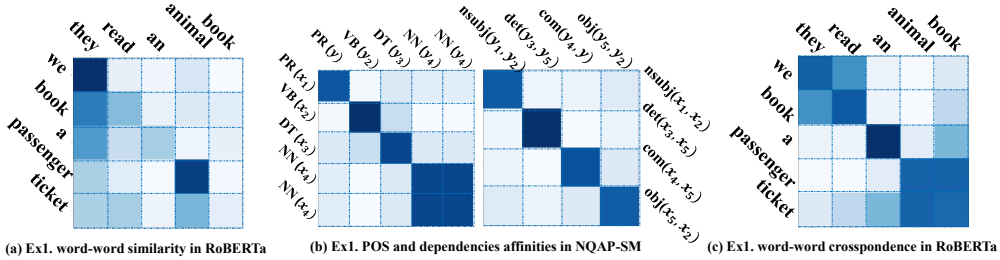


Figure 4: Cross sentence word-word similarity matrix and syntactic affinity matrices for two pairs : example 1("we book a passenger ticket","they read an animal book")

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. (2019c). Table 3 also reports the accuracy of the NQAP-SM-BERT$_{BASE}$ variation on the SciTail test data.

Compared to the original version of NQAP-SM with its 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 out-perform the BERT baseline, which indicates the effectiveness of NQAP-SM in different WAS and WRS.

Moreover, an experiment on the robustness of NQAP-SM's parameters can be found in Appendix C.

5.3.2 Matching Visualization of NQAP-SM

We conducted experiments to investigate the how the NQAP-SM matched two sentences, using two representative example sentence pairs from the aforementioned example in Figure 1 and a real example from Scitail. The experiment was conducted based on the results of NQAP-SM-RoBERTa.

Figure 4(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 4(b) illustrated the affinities between POS and dependencies in two sentences. Based on the similarities and affinity matrices, NQAP-SM solved the QAP and achieved a new correspondence matrix in Figure 4(c). The POS, word semantic similarities and dependencies affinities correspond to the node weights and edge weights in the association graph.

Example 1 illustrates the sentence pair with the similar syntax but different semantic meanings. Comparing word-word similarities by RoBERTa (Figure 4(a)) and that of by NQAP-SM (Figure 4(c)), we can see that RoBERTa's results show low similarities between words of two sentences. On other hand, NQAP-SM has the ability to align the right words because they have higher affinities (i.e. lower assignment cost). For example, bi-gram "we book" can be assigned to "they read" since they both share same POS "PR,NN" and dependencies "nsubj" shown in Figure 4(b). However, NQAM-SM will still output the "dissimilar" result due to its original low total assignment cost.

Appendix D gives another example that illustrates the sentence pair with similar semantics but different syntax.

The analysis clearly showed that NQAP-SM can utilize both the syntactic structures and semantic similarities, and make them into good interaction. Only if the sentence pair share similar syntactic and semantic information, they will be predicted as matched. The results also showed how two sentences were matched with the refined word-word similarities with the association graph.

6 Conclusion

In this paper, we present a novel sentence matching model which incorporates both syntactic and semantic matching information, referred to as NQAP-SM. NQAP-SM explicitly models the relations between syntactic and semantic information: they are used to derive association graphs. The matching of two sentences, therefore, is formalized as the optimal flow of the word correspondence over the association graphs and is solved by QAP. NQAP-SM offers several advantages: explicitly modeling the relations between syntactic and semantic matching signals, interacting both semantics and syntactic structures, and the ability in interpretation. Experimental results based on the three large-scale available benchmarks also confirmed the effectiveness, robustness, and interpretability of NQAP-SM.

References

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.

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*, pages 2–13.

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*.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 701–710.

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.

Tjalling C Koopmans and Martin Beckmann. 1957. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76.

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

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.

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.

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*, pages 1291–1299.

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.

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.

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*.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. 2019b. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7208–7215.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019c. 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.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*.

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 Conference on Research and Development in Information Retrieval*, pages 55–64.

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.

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.

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

Appendix

A A example for creating association graph

Figure 5 gives an illustrative example of creating an association graph for 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 5(a) which are represented as the node and edge weights, respectively.

Specifically, the POS tags, dependencies, and word-word similarities can be converted to the association graph (Figure 5(c)) which consists of $|X| \times |Y| = 12$ nodes and the set of nodes $\{(x, y)|x \in X, y \in Y\}$, each corresponds to a cross-sentence word-word pair sentence X and Y, respectively. The graph can also be represented with a weighted adjacency matrix, denoted as the affinity matrix $\mathbf{K}$, as shown in Figure 5(b). The weights of nodes and edges in the graph are corresponding to the diagonal and off-diagonal elements in the affinity matrix (Figure 5(b)), respectively.

The weights of nodes and edges in the graph are corresponding to the diagonal and off-diagonal elements in the affinity matrix (Figure 5(b)), respectively. Specifically, the weights of node 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.

another example is the self-loop edge to the node x_2y_2 =("likes","loves") whose weight is the semantic similarity of "likes" and "loves" plus 1 (x_2 ="likes" and y_2 ="loves" have identical POS tag "VB").

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.

Finally, the association graph corresponds to POS, syntactic dependency and word-word similarity are created. The graph can also be represented with an adjacency matrix, denoted as the affinity matrix $\mathbf{K}$, as shown in Figure 5(c).

B 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 (2) (Zhou and De la Torre, 2015):

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

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 4.1 and Section 4.2. Detailed explanations can be found Table 4.

C Robustness of NQAP-SM

NQAP-SM 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 and F_1, where $\lambda_a \in [3e-3, 1.1e-2]$. We can see that NQAP-SM performed best when $\lambda_a \approx 8e-3$. However, the performance changes were not severe (from 90.6% to 90.8% in terms of accuracy). We conclude that (1) the introduction of the affinity regularizer enables NQAP-SM to have some tolerances to the errors caused by the NLP parser, which inevitably occurs in real-world applications; (2) NQAP-SM is robust and not sensitive to the λ_a.

D Another Example for Matching Visualization

Example 2 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 the sentence pair with similar semantics but

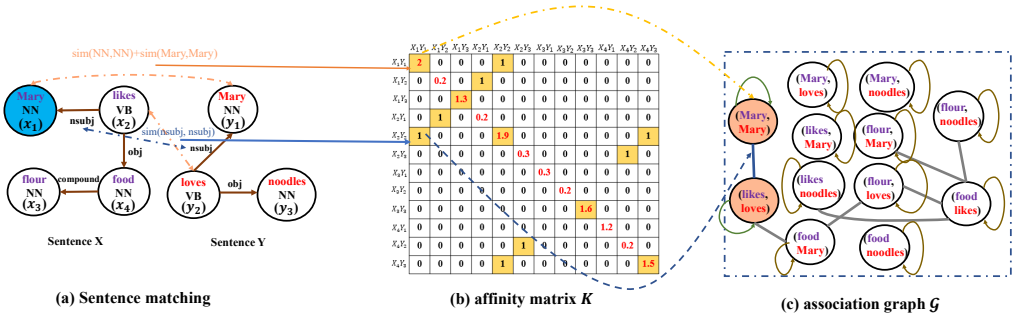


Figure 5: Example of converting the matching of a pair of sentences with syntactic structures (a) to the corresponding affinity matrix K (b) or association graph $\mathcal{G}$ (c).

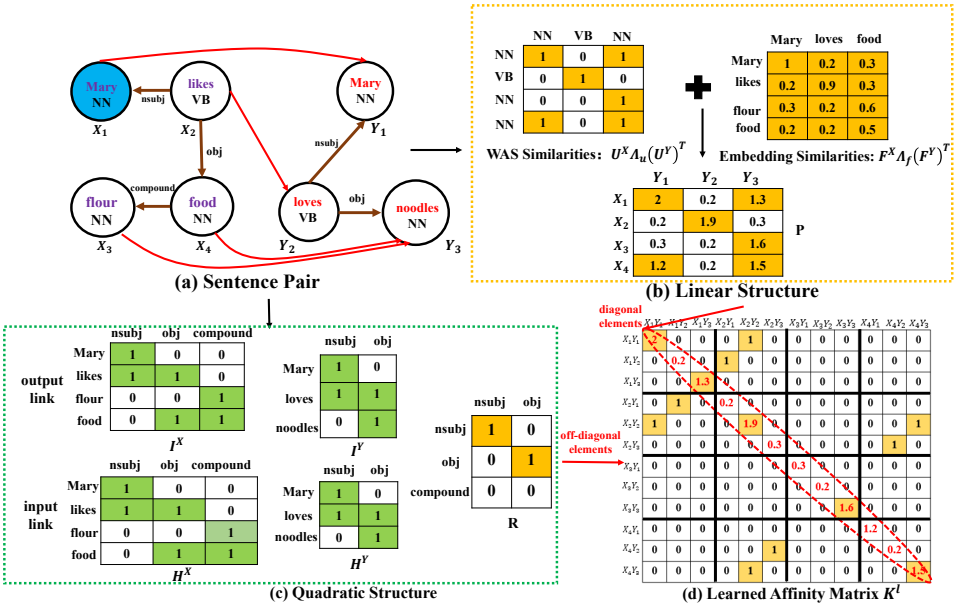


Figure 6: A working example of factorized affinity matrix K^l with aforementioned example ("Mary likes flour food", "Mary loves noodles"). The affinity matrix can be factorized into six matrices: $\mathbf{P}, \mathbf{R}, \mathbf{I}^X, \mathbf{I}^Y, \mathbf{H}^X, \mathbf{H}^Y$

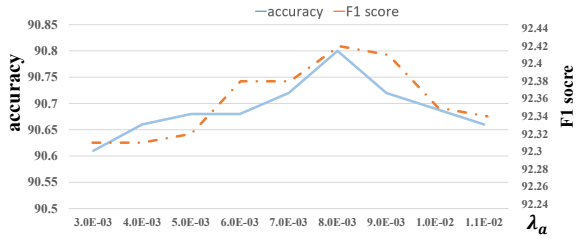
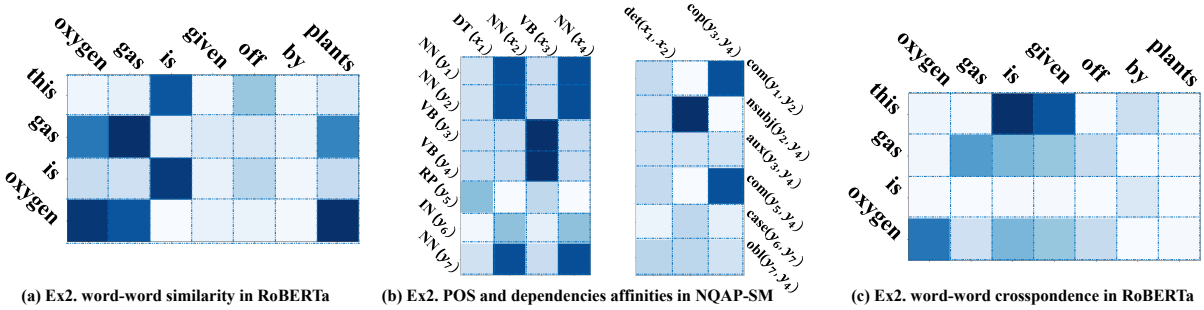


Figure 7: Accuracy and F_1 curves of NQAP-SM-BERT$_{BASE}$ w.r.t. λ_a (trade-off coefficient for affinity regularizer) on the Scitail test set.

different syntax. In Figure 8(a), we can see that the sentence pair in example 2 has high semantic similarities. However, we can see their syntactic structures are very different in Figure 8(b) and these high semantic words will have high assignment costs. For example, the dependencies "nsubj" of "gas oxygen" in X is different from the dependencies "compound" of "gas oxygen" in Y and the bi-gram will not be aligned in NQAP-SM. Finally, we can see NQAP-SM aligns a few words of X, Y in Figure 8(c) and it will generate a low total assignment cost.

Table 4: The working example of constructing $\mathbf{P}, \mathbf{R}, \mathbf{I}^X, \mathbf{I}^Y, \mathbf{H}^X, \mathbf{H}^Y$

Matrix	Formulation	Intuitive explanation with the example
$\mathbf{P}$	$\mathbf{P} = (1-\alpha)\mathbf{U}^X\mathbf{\Lambda_u}(\mathbf{U}^Y)^T + \alpha\mathbf{F}^X\mathbf{\Lambda}_f(\mathbf{F}^Y)^T$, where $\mathbf{U}^X\mathbf{\Lambda_u}(\mathbf{U}^Y)^T$ denotes the similarity matrix based on the POS tags, and $\mathbf{F}^X\mathbf{\Lambda}_f(\mathbf{F}^Y)^T$ denotes the similarity matrix based on the word embeddings.	The element $\mathbf{P}(1,1)$ (corresponding to "Mary" in X and "Mary" in Y in the example) is calculated as $\alpha \cdot 1.0 + (1-\alpha) \cdot 1.0$ where the first 1.0 denoting the semantic similarity of ("Mary","Mary") based on their word embeddings; and the second 1.0 denoting their syntactic similarity according to their POS tags ("NN","NN"), and $\alpha \in [0,1]$ is the trade-off coefficient.
$\mathbf{R}$	$\mathbf{R} = \mathbf{L}^X\mathbf{\Lambda_r}\mathbf{L}^{Y^T}$	The element $\mathbf{R}(1,1)$ (corresponding to the edge "loves -> Mary" in X and the edge "likes -> Mary" in Y in the example) in the example has the value of 1.0 because their syntactic dependency types are identical ("nsubj").
$\mathbf{I}^X, \mathbf{H}^X$	$\mathbf{I}^X(i,k) = \mathbf{H}^X(j,k) = \begin{cases} 1 & \text{if k-th edge links word} x_i \text{ to word} x_j \\ 0 & \text{otherwise,} \end{cases}$	Both $\mathbf{I}^X(2,1)$ and $\mathbf{H}^X(1,1)$ correspond to the edge "likes->Mary" in sentence X. $\mathbf{I}^X(2,1) = \mathbf{H}^X(1,1) = 1$ because the edge type is "nsubj" link from second word "likes" (x_2) to the first word "Mary" (x_1). Note that in order to reduce the noise from the dependencies, we also set a reverse edge $\mathbf{I}^X(1,1) = \mathbf{H}^X(2,1) = 1$.
$\mathbf{I}^Y, \mathbf{H}^Y$	$\mathbf{I}^Y(i,k) = \mathbf{H}^Y(j,k) = \begin{cases} 1 & \text{if k-th edge links word} y_i \text{ to word} y_j \\ 0 & \text{otherwise,} \end{cases}$	Similar to the example for $\mathbf{I}^X, \mathbf{H}^X$.



(a) Ex2. word-word similarity in RoBERTa (b) Ex2. POS and dependencies affinities in NQAP-SM (c) Ex2. word-word crosspondence in RoBERTa

Figure 8: Cross sentence word-word similarity matrix and syntactic affinity matrices for two pairs : example 2("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.