Learning Paraphrase Identification with Structural Alignment

Chen Liang[1], Praveen Paritosh[2], Vinodh Rajendran[2], and Kenneth D. Forbus[1]

[1]Northwestern University, Evanston, IL
[2]Google Research, Mountain View, CA

Abstract

Semantic similarity of text plays an important role in many NLP tasks. It requires using both local information like lexical semantics and structural information like syntactic structures. Recent progress in word representation provides good resources for lexical semantics, and advances in natural language analysis tools make it possible to efficiently generate syntactic and semantic annotations. However, how to combine them to capture the semantics of text is still an open question. Here, we propose a new alignment-based approach to learn semantic similarity. It uses a hybrid representation, attributed relational graphs, to encode lexical, syntactic and semantic information. Alignment of two such graphs combines local and structural information to support similarity estimation. To improve alignment, we introduced structural constraints inspired by a cognitive theory of similarity and analogy. Usually only similarity labels are given in training data and the true alignments are unknown, so we address the learning problem using two approaches: alignment as feature extraction and alignment as latent variable. Our approach is evaluated on the paraphrase identification task and achieved results competitive with the state-of-the-art.

1 Introduction

Semantic similarity of texts are used in many semantic tasks like paraphrase identification [Dolan and Brockett, 2005], textual entailment [Dagan *et al.*, 2005], and question answering [Voorhees, 1999]. Although simple bag-of-words models work well for large documents, short texts are challenging because those simple models would suffer from sparsity.

The semantics of text has two important parts. The first part is the local information, i.e., semantic of lexical units. Recently there has been a lot of improvements in semantic tasks using word embeddings learned from a large corpus [Baroni *et al.*, 2014; Mikolov *et al.*, 2013]. The second part is the structural information, i.e., syntactic and semantic structure. For example, a sentence pair from recently introduced Stanford NLI corpus [Bowman *et al.*, 2015] “A man wearing padded arm protection is being bitten by a German shepherd dog/A man bit a dog” is wrongly predicted as having entailment relation by both a lexicalized classifier and a sentence embedding model. If explicit syntactic and semantic structures are used, the difference between the two sentences’ meanings will be obvious. And development of natural language analysis tools [Manning *et al.*, 2014] makes it easy to efficiently generate a variety of syntactic and semantic annotations like dependency trees, POS tags, entity mentions as well as semantic role labeling and open relation extraction.

Despite the progress on both sides, how to effectively combine the local and structural information is still an open question. In this work, we propose an alignment-based approach to learn semantic similarity. It uses a hybrid representation, attributed relational graphs [Sanfeliu and Fu, 1983; Zhang and Chang, 2004], to encode lexical, syntactic and semantic information together. In order to utilize all the information, we use structural alignment as an intermediate step to support similarity estimation. Different from word alignments, structural alignment forms consistent correspondences between both words and syntactic and semantic structures of two pieces of text. To get better alignment, we introduce structural constraints to utilize the predicate-argument structure encoded in the graphs. These structural constraints are inspired by Structure Mapping Theory [Gentner, 1983], a cognitive theory of similarity and analogy.

Learning an alignment-based similarity estimator needs to overcome the challenge that usually only similarity labels are given in training data and true alignments are unknown. We investigated two approaches to deal with this problem: alignment as feature extraction and alignment as latent variable.

We evaluated our approach on the paraphrase identification task [Dolan and Brockett, 2005] and achieved results competitive with the state-of-the-art.

2 Related Work

A lot of different approaches have been proposed for text similarity. We focus on two main categories most related to ours.

Neural network models extend the idea of word embedding to larger constructions like phrases, sentences and paragraphs. A variety of architectures have been explored. The recursive neural network in [Socher *et al.*, 2011] used a constituency tree and recursive autoencoder to learn composition functions of word embeddings to phrase embeddings and eventually sentence embeddings. Tree-LSTM [Tai *et al.*, 2015] generalizes LSTM from sequences to tree structures. Other recent approaches include Siamese architecture with syntax-aware multi-sense embeddings [Cheng and Kartsaklis, 2015] and convolutional neural networks [He *et al.*, 2015; Yin and Schütze, 2015]. Despite strong performance, distributed representations are usually less interpretable than explicit structured representations, and it is unclear whether fixed length vectors are expressive enough to represent all linguistic structures. Our approach also uses word embeddings, but instead of trying to compress all the information into one fixed length vector, we use word embeddings together with other linguistic structures explicitly encoded in the attributed relational graph.

Some approaches explore syntactic structures of two sentences. Tree kernels and graph kernels with SVM have been used on syntactic structures extended with relational links between matching words [Filice *et al.*, 2015] to predict relations between texts. Quasi-synchronous grammar was used in [Das and Smith, 2009] to model divergence of syntactic structure between paraphrases. [Beltagy *et al.*, 2014] used probabilistic soft logic to combine logical and distributional representations through weighted inference rules. Our method takes a similar hybrid approach, but uses attributed relational graphs as an extensible representation to encode various different kinds of lexical, syntactic and semantic information.

Alignment has been used as an intermediate step in several NLP tasks. For example, word alignment as latent variable was used in machine translation [Liang *et al.*, 2006]. Neural attention models in machine translation [Bahdanau *et al.*, 2014] and textual entailment [Rocktäschel *et al.*, 2015] can be seen as jointly learning soft alignments between words in source and target sentences or words in text and hypothesis. The problem that alignments are latent in data is a common challenge for these alignment-based approaches. Latent variable model and joint learning is a commonly used solution. We take a similar approach here. However, different from word alignment, our alignment is carried on both words and syntactic structures. By using structural constraints, it utilizes the predicate-argument structure in the text, which is generally ignored in other works. [Sammons *et al.*, 2009] and [Chang *et al.*, 2010] used alignment of structured annotations from different resources for textual entailment and paraphrase identification. This work follows a similar approach, but shows that using simpler alignment and learning methods and adding word embeddings as another resource can achieve state-of-the-art performance on paraphrase identification.

Structural alignment as an intermediate step to support similarity estimation also has support from cognitive science. Structure Mapping Theory (SMT) [Gentner, 1983] states that similarity judgment and analogy is done through structural alignment of mental representations. In the alignment process, humans prefer structurally consistent alignment of deep nested structures, which is called structural consistency and systematicity principle. This theory and its computational model Structure Mapping Engine (SME) [Falkenhainer *et al.*, 1989] has been proven to fit human performance in various different experiments [Forbus, 2001]. A variation of the algorithm was used in the IBM Watson Jeopardy system for evaluating candidate answers [Murdock, 2011]. Recently [Liang and Forbus, 2015] combined SME with statistical learning and showed state-of-the-art performance on knowledge base completion task using orders of magnitude less training data than other approaches. Structural alignment is a major component of our approach, and the structural constraints we introduced to our alignment model is inspired by SMT's structural consistency principle.

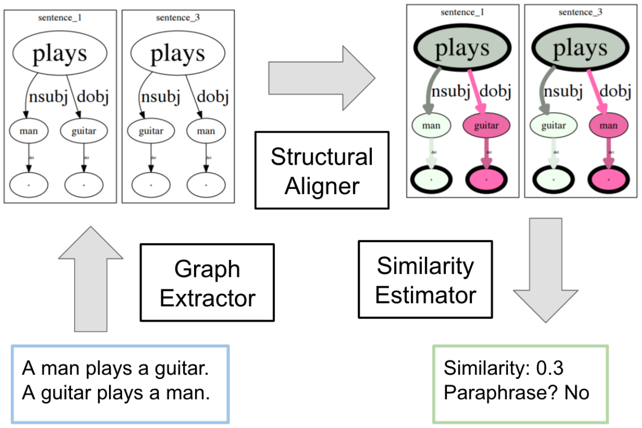

Figure 1: An overview of the full pipeline consisting of three main components to turn raw texts into attributed relational graphs, structurally align them and estimate their similarity. The color indicates how the nodes and edges from two graphs match with each other in the alignment.

3 Method

The problem of semantic similarity is, given two pieces of text, the system must produce a score indicating the degree that their meanings are equivalent. In this work, we focus on a special case, paraphrase identification, in which the system predicts whether two sentences can be considered semantically equivalent or not. We solve this problem with a pipeline of three components similar to RATER [Sammons *et al.*, 2009], as shown in Figure 1.

1. **Graph extractor**: Given two pieces of text, it uses word embeddings and a set of automatic annotators to extract the tokens, syntactic relations, POS tags and entity mentions to generate the attributed relational graphs.
2. **Structural aligner**: Given two attributed relational graphs, the structural aligner generates an alignment. An alignment of two attributed relational graphs is a set of matches, and each match is a correspondence between two nodes or edges.
3. **Similarity estimator**: Given an alignment, the similarity estimator produces a similarity score between the two graphs or a label indicating whether they are similar enough to be considered equivalent or not.

3.1 Data Representation & Preprocessing

Our method uses a hybrid representation, attributed relational graphs (directed graphs with attributes attached to the nodes and edges). The attributes store local information about a unit/node or a relation/edge, which will later be used to extract features for each match between two nodes or two edges. These features are used to estimate the similarity and importance of the match. In our experiment, for fair comparison with other methods, we just use tokens as units/nodes and dependency arcs as relations/edges. For attributes, we used dependency label, token, lemma, POS tag, NER tag and word embedding. These annotations can all be obtained easily through standard natural language analysis tools. Here we used Stanford CoreNLP and pretrained Word2Vec word embeddings[1] [Mikolov *et al.*, 2013].

For example, as shown in Figure 2, given a sentence "A man plays a guitar", the graph extractor would first get tokens, lemmas, dependency tree, POS tags and NER tags from Stanford CoreNLP, and word embeddings from pretrained Word2Vec model. Then it creates one node for each token and one edge for each dependency arc[2] connected to the two nodes that correspond to its head and dependent token. The annotations of each token, like its lemma, POS and NER tag and its word embedding would be attached to the corresponding node as attributes, and the dependency label of the dependency arc will be attached to the edge as an attribute.

There are two advantages of this representation. First, it is expressive enough to encode heterogeneous structural information in the same graph; second, local information can be easily encoded as attributes. This makes the representation easy to extend. New structural annotations such as semantic role labeling or relation extraction and new lexical or phrasal resources can be easily added to the representation. Compared to finite length feature vectors that inevitably lose some structural information, this representation preserves the structure explicitly. Compared to logic or purely symbolic representations, this representation can easily include different kinds of features to train feature-rich discriminative models for learning alignments and similarity estimation from data.

3.2 Structural Alignment & Similarity Estimation

The two core components of our approach are the *structural aligner* and *similarity estimator*. Given two input graphs, the structural aligner finds the best alignment between them, which can be seen as a structured prediction problem. Based on the best alignment, the similarity estimator produces a score indicating the degree of similarity or a binary output indicating whether the two sentences are semantically equivalent or not, which can be seen as a regression or classification task depending on which output is produced.

An alignment is a set of matches. Each match is a pair of nodes or edges from the two graphs. The structural alignment has two steps. First, given two graphs, the structural aligner generates all the possible matches that pass some criteria. In this work, the criteria we used are that (1) two matched dependency arcs must have the same dependency label; (2) the cosine similarity between word embeddings of two matched tokens must be greater than 0.4. This value is chosen from pilot experiments with a subset of the data. The result is not sensitive to it because most of the matches that passed this threshold have similarities much higher than it. Second, it selects the subset of matches that optimizes an objective defined in Equation 1.

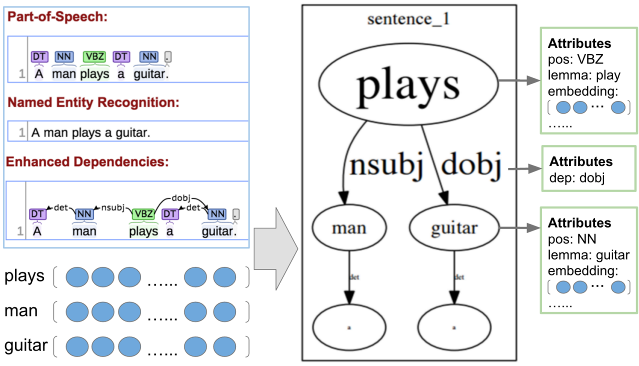

Figure 2: A example of annotations turned into an attributed relational graph, structural information such as syntactic structure is encoded in the graph structure, and local information such as word embedding or dependency label is encoded in the attributes attached to nodes and edges

Formalized as a structured prediction problem, the input x is a pair of graphs and the output a is an alignment. Φ_a(x, a) is a function mapping input graphs and a candidate alignment to a feature vector. The features used will be discussed in Section 3.3. Let w_a be the set of parameters for the structural aligner, and A be all the possible subsets of all the possible matches. Finding the best alignment is solving the following problem:

$$a_{pred} = \underset{a \in A}{\text{argmax}}\, w_a \cdot \Phi_a(x, a) \qquad (1)$$

This argmax problem is intractable, so we use beam search to find an approximate solution.

Formalized as a regression or classification problem, the similarity estimator takes the pair of graphs and the predicted alignment as input, and uses another feature function $\Phi_s(x, a_{pred})$ to map them to a feature vector. Let w_s be the set of parameters for the similarity estimator. Here we consider the paraphrase identification task. Since the correct similarity label y_{true} is usually given in training data, this is a supervised binary classification problem. Using a linear classifier, the similarity label y_{pred} is predicted as:

$$y_{pred} = sgn(w_s \cdot \Phi_s(x, a_{pred})) \qquad (2)$$

In this work, we use SVM to learn w_s.

However, learning w_a is more challenging because the true alignment a_{true} is usually latent. We address this problem using two approaches, alignment as feature extraction and alignment as latent variable.

In the first approach, we consider the structural aligner as a feature extractor and w_s as hyperparameters, and use grid search over a validation set to select the best parameters. But the problem is that the number of runs needed in grid search grows exponentially with the number of hyperparameters, So we have to restrict Φ_s to include just a small set of features.

[1] https://code.google.com/archive/p/word2vec/

[2] The root dependency arc is encoded as an attribute of the node of the root token instead of an edge.

In addition, the grid search cannot be very fine-grained due to its computational cost.

To overcome these problems and utilize more features in the structural aligner, the second approach jointly trains the structural aligner and the similarity estimator through an iterative process, as follows. First, we initialize the parameters to some value w_a^0, for example using the values obtained from the first approach. Then we repeat two steps:

(1) Keep w_a fixed, for each input x^i, assume the alignments produced by the structural aligner is "correct" and learn w_s from examples $\{(x^i, a^i_{pred}, y^i_{true}), i = 1, 2 \ldots, n\}$.

(2) Keep w_s fixed, for each input x^i, hallucinate the "correct" alignment a^i_* by solving

$$a^i_* = \operatorname*{argmax}_{a \in A^i} w_a \cdot \Phi_a(x^i, a) - C \cdot Loss(y_{true}, w_s \cdot \Phi_s(x^i, a)) \tag{3}$$

where C is a hyperparameter that represents the confidence in the classifier. Since we use SVM to learn w_s, the $Loss$ here is hinge loss. In the experiment, we set C to be very large (10^6). Thus, this $argmax$ solves for the alignment that causes the lowest prediction error and, if there is a tie in the error, has the highest alignment score $w_a \cdot \Phi_a(x^i, a)$. Then, examples $\{(x^i, a^i_*), i = 1, 2 \ldots, n\}$ are used to train the aligner using the averaged structured perceptron algorithm [Collins, 2002].

Because this process will usually overfit the training data, we use the performance on a validation set to decide when to stop.

3.3 Features

In this section, we discuss the features used in feature functions Φ_a and Φ_s, and focus on how the pairwise features are used to encode the structural constraints. We first describe a feature function Φ, then show how Φ_a and Φ_s are defined using Φ.

The (global) feature vector $\Phi(x, a)$ is a concatenation of global unary feature vector $\Phi_u(x, a)$ and global pairwise feature vector $\Phi_p(x, a)$.

$$\Phi(x, a) = [\Phi_u(x, a);\ \Phi_p(x, b)] \tag{4}$$

The global unary and pairwise feature vectors are aggregations of local unary and pairwise feature vectors. In other words, a global feature vector is just a sum of local feature vectors. Recall that an alignment a is just a set of individual matches $\{m_i\}$. The local unary feature vector $\phi_u(x, m_i)$ is computed for each individual match m_i, and the local pairwise feature vector $\phi_p(x, m_i, m_j)$ is computed for each pair of matches (m_i, m_j).

$$\Phi_u(x, a) = \sum_i \phi_u(x, m_i) \tag{5}$$

$$\Phi_p(x, a) = \sum_{i,j} \phi_p(x, m_i, m_j),\ i \neq j \tag{6}$$

For the structural aligner, only the ranking of candidate alignments matters, because it just produces the one with highest score as the output. In this case, the aggregation of local features Φ suffices to be a good feature vector. So Φ_a is simply defined as:

$$\Phi_a(x, a) = \Phi(x, a) \tag{7}$$

For the similarity estimator, however, the value of $w_s \cdot \Phi_s(x, a_{pred})$ matters because its sign decides the prediction and its absolute value roughly represents the confidence in the prediction. So the feature vector needs to be normalized.

Here we use the self alignments to normalize. Note that the input x is just a pair of attributed relational graphs (g_1, g_2) extracted from the pair of sentences. The self alignment feature vector is defined as:

$$\Phi_{self}(x) = \frac{\Phi(x^1_{self}, a^1_{self}) + \Phi(x^2_{self}, a^2_{self})}{2} \tag{8}$$

where $x^1_{self} = (g_1, g_1)$ and $x^2_{self} = (g_2, g_2)$. a^1_{self} and a^2_{self} are the self alignments of g_1 and g_2, in which each node and edge just matches to itself.

Then Φ_s is defined as concatenation of three vectors:

$$\Phi_s(x, a) = [\Phi(x, a);\ \Phi_{self}(x);\ \frac{\Phi_{self}(x) - \Phi(x, a)}{\Phi_{self}(x) + \delta}] \tag{9}$$

where δ is a very small smoothing term. The third vector is the normalized difference between self alignment's aggregation feature vector and the current alignment a's aggregation feature vector. Because each dimension i of the vector corresponds to one feature and for most of the features[3] we used, $0 < \Phi(x, a)_i < \Phi_{self}(x)_i$. So each dimension of the third term is bounded between 0 and 1. We still include the raw aggregation feature vector $\Phi(x, a)$ and self alignment aggregation feature vector $\Phi_{self}(x)$ in the final feature vector because the raw values of these features are also informative and were shown to improve the performance in the experiments.

Unary Features

Unary features are used to estimate how similar two matched tokens or dependency arcs are, and also how important they are in their sentences. These features are used to compute how much this match will contribute to the alignment score or overall similarity. Listed below are all the unary features we used in this work.

1. **Lexical similarity**: cosine similarity between word embeddings of the matched two tokens.
2. **Lexical features**: word features for words that appeared at least twice, lemma features and an indicator feature for whether the two matched tokens have the same lemma.
3. **Syntactic features**: POS tag features, and an indicator feature of whether the matched two token has the same POS tag; dependency label features, and an indicator feature of whether the matched two dependency arcs have the same dependency label.
4. **NER features**: NER tag features, and two indicator features, one for whether the matched two tokens has the same NER tag, and the other for whether the matched two tokens have the same normalized entity name.
5. **Position difference feature**: the absolute difference between the positions of the matched two tokens in the their sentences.

[3] The only feature that violates this is the position difference feature, so we don't include it in this normalized term and just keep its raw values.

Pairwise Features

Pairwise features are introduced to improve alignment by encoding the structural constraints between matches. These structural constraints ensure that the final alignment is structurally consistent. They are inspired by the structural consistency principle of Structure Mapping theory. The principle states that two constraints are used by human when aligning predicate-argument structures: (1) one-to-one mapping that one entity or predicate should only match to one entity or predicate; (2) parallel connectivity that if a predicate matches another predicate, their roles and arguments should also match correspondingly.

In this work, we adapted these constraints to work on tokens and syntactic relations. The one-to-one mapping is encoded as a hard constraint in the structural aligner so that alignments that matches one token or relation to more than one other token or relation would be filtered out. The parallel connectivity constraint is adapted by considering dependency tree as an approximate predicate-argument structure. So if the heads of two dependency arcs match, the two dependency arcs should also be more likely to match. If two dependency arcs match, the dependent of the dependency arcs should be more likely to match as well.

These two constraints are implemented as two pairwise indicator features. For a match m_i between two tokens t_a and t_b, and a match m_j between two dependency arcs d_a and d_b:

$$\phi_p(x, m_i, m_j)_1 = \begin{cases} 1, & \text{if } t_a = head(d_a) \;\&\; t_b = head(d_b) \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$\phi_p(x, m_i, m_j)_2 = \begin{cases} 1, & \text{if } t_a = dep(d_a) \;\&\; t_b = dep(d_b) \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

dep is short for $dependent$.

Note that in this work, for simplicity, we just used pairwise features to demonstrate the utility of structural constraints. Structural constraints involving more matches can be modeled using higher-order features, and more fine-grained constraints can make use of the attributes of nodes and edges.

4 Experiment

We evaluated our model on the paraphrase identification task, a benchmark for evaluating many semantic similarity models.

4.1 Dataset

The dataset is MSRP paraphrase corpus [Dolan and Brockett, 2005]. It contains 5801 pairs of sentences, each labeled with a binary judgment indicating whether human raters considered the pair of sentences to be semantically similar enough to be considered paraphrases. Among the sentences, 67% are positive examples. As is described in the paper, "the majority of the 'equivalent' pairs in this dataset exhibit 'mostly bidirectional entailments', with one sentence containing information that differs from or is not contained in the other". Due to its construction method, the lexical overlap between two sentences in a pair are usually very high. We used the training and test split provided by the corpus (4076 training and 1725 test examples), and selected 20% of the training data by stratified sampling as the validation set.

Model	Accuracy	F1
Das and Smith (2009)	76.1%	82.7%
Wan et al. (2006)	75.6%	83.0%
Socher et al. (2011)	76.8%	83.6%
Madnani et al. (2012)	77.4%	84.1%
He et al. (2015)	78.6%	84.7%
Filice et al. (2015)	79.1%	85.2%
Cheng and Kartsaklis (2015)*	78.6%	85.3%
Ji and Eisenstein (2013)*	80.4%	85.9%
This work		
baseline	74.6%	82.6%
local similarity	76.9%	83.9%
+structural constraints	77.4%	84.2%
+syntactic features	78.2%	84.7%
+latent variable model	78.3%	84.9%

Table 1: Test results on MSRP paraphrase dataset. The results labeled with * used extra data besides the given training set (discussed more in Section 4.2).

4.2 Results & Analysis

We used different experiment settings to analyze the contributions of different components, and compared our model to other state-of-the-art models[4]. The result is shown in Table 1.

Despite its simplicity, our model is competitive to the state-of-the-art. The two results labeled with * used extra data besides the training set. [Ji and Eisenstein, 2013] used matrix factorization on both training and test set to extract distributional features, and [Cheng and Kartsaklis, 2015] used PPDB [Ganitkevitch *et al.*, 2013], which is several orders of magnitude larger, as training data. [Filice *et al.*, 2015] used trees as structured representations of text, and lexical matching was also an important component in their method. They achieved better results with more complex features and a thorough comparison and combination of different graph and tree kernels, while our much simpler alignment-based method showed comparable performance.

For different experiment settings, we kept the features used in the similarity estimator fixed, and varied the features used in the aligner. For the first three settings, the parameters for the aligner are decided using grid search over a validation set.

The **baseline** setting uses SVM with a set of simple features: (1) cosine similarity between average word embeddings of the two sentences; (2) simple number features, percentage of words in the other sentence and sentence length difference used in [Socher *et al.*, 2011]; (3) BLEU1 through BLEU4 as separate features.

The **local similarity** setting uses just lexical similarity and same dependency label features. In other words, it uses only local similarity of the matched two tokens or dependency arcs to find the best alignment. Surprisingly, this simple approach already outperforms the baseline and three other sophisticated models. This showed that word embeddings and shallow features are not good enough, but the combination of word embeddings with alignment and rich features is surprisingly effective.

[4] `http://aclweb.org/aclwiki/index.php?title=Paraphrase_Identification_(State_of_the_art)`

Using only local similarity, the alignment contains separating matches that do not connect with each other. The **+structural constraints** setting added the two pairwise indicator features to promote structural consistency. With these two features as structural constraints, the aligner prefers a set of consistent matches between two connected syntactic tree structure over matches between scattered pieces. This improved the F1 score by another 0.3 percent.

Since the dependency tree is used as an approximation to the predicate-argument structure, this approximation makes more sense for some dependency relations, such as *nsubj*, *nsubjpass* and *dobj*, and some words, such as verbs and nouns, than others. Using this heuristic, the **+syntactic features** setting added verb and noun POS tag features and features for those three dependency labels. This would help the aligner focus on the words and dependency relations that capture the predicate-argument structure better during the search for a set of structurally consistent matches. This increased the F1 score by another 0.4 percent.

The **+latent variable model** setting explored the full parameter space for the aligner using the iterative training described in Section 3.2. It takes the best parameters from last setting as initialization. We run averaged structured perceptron for 10 epochs. The averaged parameters after each epoch is stored as $\{w_i, i = 1, 2, ..., 10\}$, and we used a validation set to decide which one to use as the final parameters. We also used the validation set to decide when to stop the iterative training. This further improved the F1 score by another 0.2 percent.

We used 5-fold cross validation on the whole dataset to check the statistical significance of the differences between settings. We found that the improvement of local similarity setting to the baseline and the improvement of the full model to the local similarity setting are both statistically significant. But the differences within the three structural alignment settings are more subtle and not statistically significant.

The alignment and rich features enabled the system to learn which part of the sentences are more important to its semantic rather than treating them all the same. This eliminates many false positives caused by misleading lexical overlap. For example, "Gyorgy Heizler, head of the local disaster unit, said the coach had been carrying 38 passengers.", and "The head of the local disaster unit, Gyorgy Heizler, said the coach driver had failed to heed red stop lights." was classified wrongly as paraphrase by the baseline because of high lexical overlap, but local similarity setting made the right prediction.

Structural alignment further eliminates false positives because it helps constrain the lexical matches. For example, "a dog bites a man" and "a man bites a dog" have a perfect alignment in local similarity setting, but will not be recognized as similar by the full model, because the syntactic structures lead the aligner to match "dog" with "man" and "man" with "dog", which are not similar.

4.3 Discussion & Future Work

The results demonstrated the utility of the hybrid representation, attributed relational graphs. It enables the integration of two processes: (1) similarity estimation, which uses a strong classifier based on aggregations of local features; (2) structural alignment, which utilizes the structure to extract a structurally consistent set of local features. Another major advantage of using attributed relational graph is that it can be extended with other local or structural resources easily. One interesting future variation is to incorporate more structures, such as relations from semantic role labeling and open relation extraction, as well as more units, such as phrases and linked entities, into the attributed relational graph.

In addition, the results also showed the effectiveness of alignment as feature extraction, and the utility of the two structural constraints when the predicate-argument structure (approximately) exists in the input. A next step is to test this approach on other semantic NLP tasks like textual entailment and question answering.

Moreover, learning the parameters for alignment from data would be important for adapting to different datasets and tasks. It would also increase the capacity of the model. The improvement from using alignment as latent variable on this dataset is limited for two reasons: (1) the alignment problem is not so difficult due to the high lexical overlap of the pairs of sentences in the corpus, so some simple features could already provide high quality alignments; (2) the dataset is not large enough, so it is easy to overfit with the joint learning. Thus, scaling up to large datasets and testing on corpus with less lexical overlap and more structural differences would be another next step, and might require integration with compositional models.

5 Conclusion

In this work, we proposed a new alignment-based approach to learn semantic similarity of texts and evaluated it on the paraphrase identification task. We used a hybrid representation, attributed relational graphs, to encode local and structural information. This enables us to integrate structural alignment and similarity estimation through two approaches: alignment as feature extraction and alignment as latent variable. To improve the alignment, we also introduced two structural constraints that make use of the predicate-argument structures. In the experiment, our approach achieved results competitive with other state-of-the-art models on the MSRP corpus. Further analysis showed the strength of this hybrid approach and confirmed contributions of structural alignment using structural constraints and joint learning.

Acknowledgments

We thank Thanapon Noraset for helpful comments on the draft of the paper.

References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[Baroni *et al.*, 2014] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, 2014.

[Beltagy *et al.*, 2014] Islam Beltagy, Katrin Erk, and Raymond J. Mooney. Probabilistic soft logic for semantic textual similarity. In *ACL*, 2014.

[Bowman *et al.*, 2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

[Chang *et al.*, 2010] Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. Discriminative learning over constrained latent representations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 429–437. Association for Computational Linguistics, 2010.

[Cheng and Kartsaklis, 2015] Jianpeng Cheng and Dimitri Kartsaklis. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *EMNLP*, 2015.

[Collins, 2002] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

[Dagan *et al.*, 2005] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *MLCW*, 2005.

[Das and Smith, 2009] Dipanjan Das and Noah A. Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *ACL*, 2009.

[Dolan and Brockett, 2005] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*, 2005.

[Falkenhainer *et al.*, 1989] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine: Algorithm and examples. *Artif. Intell.*, 41:1–63, 1989.

[Filice *et al.*, 2015] Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. Structural representations for learning relations between pairs of texts. In *ACL*, 2015.

[Forbus, 2001] K Forbus. *Exploring analogy in the large*. MIT Press, 2001.

[Ganitkevitch *et al.*, 2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *NAACL*, 2013.

[Gentner, 1983] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155–170, 1983.

[He *et al.*, 2015] Hua He, Kevin Gimpel, and Jimmy Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*, 2015.

[Ji and Eisenstein, 2013] Yangfeng Ji and Jacob Eisenstein. Discriminative improvements to distributional sentence similarity. In *EMNLP*, 2013.

[Liang and Forbus, 2015] Chen Liang and Kenneth D. Forbus. Learning plausible inferences from semantic web knowledge by combining analogical generalization with structured logistic regression. In *AAAI*, 2015.

[Liang *et al.*, 2006] Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Benjamin Taskar. An end-to-end discriminative approach to machine translation. In *ACL*, 2006.

[Manning *et al.*, 2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, 2014.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[Murdock, 2011] J. William Murdock. Structure mapping for jeopardy! clues. In *ICCBR*, 2011.

[Rocktäschel *et al.*, 2015] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomá s Kociský, and Phil Blunsom. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664, 2015.

[Sammons *et al.*, 2009] Mark Sammons, V. G. Vinod Vydiswaran, Tim Vieira, Nikhil Johri, Ming-Wei Chang, Dan Goldwasser, Vivek Srikumar, Gourab Kundu, Yuancheng Tu, Kevin Small, Joshua Rule, Quang Do, and Dan Roth. Relation alignment for textual entailment recognition. In *TAC*, 2009.

[Sanfeliu and Fu, 1983] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:353–362, 1983.

[Socher *et al.*, 2011] Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, 2011.

[Tai *et al.*, 2015] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015.

[Voorhees, 1999] Ellen M. Voorhees. The TREC-8 question answering track report. In *TREC*, 1999.

[Yin and Schütze, 2015] Wenpeng Yin and Hinrich Schütze. Convolutional neural network for paraphrase identification. In *NAACL*, 2015.

[Zhang and Chang, 2004] DongQing Zhang and Shih-Fu Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *MM*, 2004.