Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering

Mengqiu Wang
Computer Science Department
Stanford University
mengqiu@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
manning@cs.stanford.edu

Abstract

A range of Natural Language Processing tasks involve making judgments about the semantic relatedness of a pair of sentences, such as Recognizing Textual Entailment (RTE) and answer selection for Question Answering (QA). A key challenge that these tasks face in common is the lack of explicit alignment annotation between a sentence pair. We capture the alignment by using a novel probabilistic model that models tree-edit operations on dependency parse trees. Unlike previous tree-edit models which require a separate alignment-finding phase and resort to ad-hoc distance metrics, our method treats alignments as structured latent variables, and offers a principled framework for incorporating complex linguistic features. We demonstrate the robustness of our model by conducting experiments for RTE and QA, and show that our model performs competitively on both tasks with the same set of general features.

1 Introduction

Many complex Natural Language Processing (NLP) applications can be broken down to a subtask of evaluating the semantic relationship of pairs of sentences (e.g., in Question Answering, answer selection involve comparing each answer candidate against the question). This means that research aiming at analyzing pairs of semantically related natural language sentences is promising because of its reusability: it is not tied to a particular internal representation of meanings, but it nevertheless serves as a first step towards full meaning understanding, which is applicable to a number of applications. At the same time, this paradigm clearly defines the input and output space, facilitating system comparison and standard evaluation. Tasks of this paradigm have drawn much of the focus in recent NLP research, including Recognizing Textual Entailment (RTE), answer selection for Question Answering (QA), Paraphrase Identification (PI), Machine Translation Evaluation (MTE), and many more.

In each of these tasks, inputs to the systems are pairs of sentences that may or may not convey the desired semantic property (e.g., in RTE, whether the hypothesis sentence can be entailed from the premise sentence; in QA, whether the answer candidate sentence correctly answers the question), and the output of the system is a binary classification decision (or a regression score,as in MTE).

Earlier studies in these domains have concluded that simple word overlap measures (e.g., bag of words, n-grams) have a surprising degree of utility (Papineni et al., 2002; Jijkoun and de Rijke, 2005b), but are nevertheless not sufficient for these tasks (Jijkoun and de Rijke, 2005a). A common problem identified in these earlier systems is the lack of understanding of the semantic relation between words and phrases. Later systems that include more linguistic features extracted from resources such as WordNet have enjoyed more success (MacCartney et al., 2006). Studies have also shown that certain prominent syntactic features are often found beneficial (Snow et al., 2006). More recent studies gained further leverage from systematic exploration of the syntactic feature space through analysis of parse trees (Wang et al.,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1164–1172,
Beijing, August 2010

2007; Das and Smith, 2009).

There are two key challenges imposed by these tasks. The first challenge has to do with the hidden alignment structures embedded in the sentence pairs. It is straightforward to see that in order to extract word-matching and/or syntax-matching features, inevitably one has to consider the alignment between words and/or syntactic parts. These alignments are not given as inputs, and it is a non-trivial task to decide what the *correct* alignment is. Alignment-based approach have been proven effective by many RTE, QA and MTE systems (Haghighi et al., 2005; Wang et al., 2007; MacCartney et al., 2008; Das and Smith, 2009, *inter alia*). Although alignment is a commonly used approach, it is not the only one. Other studies have successfully applied theorem proving and logical induction techniques, translating both sentences to knowledge representations and then doing inference on these representations (Moldovan et al., 2003; Raina et al., 2005; de Salvo Braz et al., 2005; MacCartney and Manning, 2007, *inter alia*).

A second challenge arises when a system needs to combine various sources of evidence (i.e., surface text features, semantic features, and syntactic features) to make a global classification decision. Quite often these features are heavily overlapping and sometimes contradicting, and thus a robust learning scheme that knows when to activate what feature is desired. Traditional approaches employ a two-stage or multi-stage model where tasks are broken down into alignment finding, feature extraction, and feature learning subtasks (Haghighi et al., 2005; MacCartney et al., 2008). The alignment finding task is typically done by committing to a *one best* alignment, and subsequent features are extracted only according to this alignment. A large body of literature in joint learning has demonstrated that such an approach can suffer from cascaded errors at testing, and does not benefit from the potential for joint learning (Finkel et al., 2006).

In this paper, we present a novel undirected graphical model to address these challenges. A promising approach to these challenges is modeling the alignment as an edit operation sequence over parse tree representation, an approach pioneered by (Punyakanok et al., 2004; Kouylekov and Magnini, 2006; Harmeling, 2007; Mehdad, 2009). We improve upon this earlier work by showing how alignment structures can be inherently learned as structured latent variables in our model. Tree edits are represented internally as state transitions in a Finite-State Machine (FSM), and our model is parameterized as a Conditional Random Field (CRF) (Lafferty et al., 2001), which allows us to incorporate a diverse set of arbitrarily overlapping features.

In comparison to previous work that exploits various ad-hoc or heuristic ways of incorporating tree-edit operations, our model provides an elegant and much more principled way of describing tree-edit operations in a probabilistic setting.

2 Tree-edit CRF for Classification

A training instance consists of a pair of sentences and an associated binary judgment. In RTE, for example, the input sentence pairs is made up of a text sentence (e.g., *Gabriel Garcia Marquez is a novelist and winner of the Nobel prize for literature.*) and a hypothesis sentence (e.g., *Gabriel Garcia Marquez won the Nobel for Literature.*). The pair is judged to be *true* if the hypothesis can be entailed from the text (e.g., the answer is *true* for the example sentence pair).

Formally, we denote the text sentence as *txt* and the hypothesis sentence as *hyp*, and denote their labeled dependency parse trees as $\tau_{\mathbf{t}}$ and $\tau_{\mathbf{h}}$, respectively. We use the binary variable $z \in \{0,1\}$ to denote the judgment.

The generative story behind our model is a parse tree transformation process. $\tau_{\mathbf{t}}$ is transformed into $\tau_{\mathbf{h}}$ through a sequence of *tree edits*. Examples of tree edits are *delete child*, *insert parent*, and *substitute current*. An edit sequence $\mathbf{e} = e_1 \ldots e_m$ is *valid* if $\tau_{\mathbf{t}}$ can be successfully turned into $\tau_{\mathbf{h}}$ according to $\mathbf{e}$. An example of a trivial valid edit sequence is one that first *deletes* all nodes in $\tau_{\mathbf{t}}$ then *inserts* all nodes in $\tau_{\mathbf{h}}$.

Delete, *insert* and *substitute* form the three basic edit operations. Each step in an edit sequence is also linked with current edit positions in both trees, denoted as $\mathbf{e}.\mathbf{p} = e_1.p \ldots e_m.p$. We index the tree nodes using a level-order tree traversal scheme (i.e., root is visited first and assigned in-

dex 0, then each one of the first level children of the root is visited in turn, and assigned an index number incremented by 1). It is worth noting that every valid edit sequence has a corresponding alignment mapping. Nodes that are inserted or deleted are aligned to *null*, and nodes that are substituted are aligned. One can find many edit sequence for the same alignment, by altering the order of edit operations.

We extend these basic edit operations into more elaborate edit operations based on the linguistic and syntactic properties of the current tree nodes that they fire on. For example, the following are all possible edit operations: *delete* a noun that is SUB of the root, *delete* a named-entity of type PERSON, *substitute* roots of the tree. In our experiments, we designed a set of 45 edit operations (12 *delete*, 12 *insert* and 21 *substitute*). More details of the edit operations are described in §4. Depending on the specific application domain, more sophisticated and verbose tree edit operations can be designed and easily incorporated into our model. In particular, tree edit operations involving deleting, inserting or substituting entire treelets seem interesting and promising, requiring merely a simple extension to the forward-backward dynamic programming.

Next, we design a Finite-State Machine (FSM) in which each edit operation is mapped to a unique state, and an edit sequence is mapped into a transition sequence among states (denoted as $\mathbf{e}.\mathbf{a} = e_1.a \ldots e_m.a$). In brief, an edit sequence is associated with a sequence of edit positions in the trees ($\mathbf{e}.\mathbf{p} = e_1.p \ldots e_m.p$), as well as a transition sequence among states ($\mathbf{e}.\mathbf{a} = e_1.a \ldots e_m.a$).

The probability of an edit sequence $\mathbf{e}$ given the parse trees is defined as:

$$P(\mathbf{e} \mid \tau_\mathbf{t}, \tau_\mathbf{h}) = \frac{1}{Z} \prod_{i=1}^{|\mathbf{e}|} \exp \theta \cdot \mathbf{f}(e_{i-1}, e_i, \tau_\mathbf{t}, \tau_\mathbf{h}) \quad (1)$$

where $\mathbf{f}$ are feature functions, θ are associated feature weights, and Z is the partition function to be defined next.

Recall that our training data is composed of not only positive examples but also negative examples. In order to take advantage of this label information, we adopt an interesting discriminative learning framework first introduced by McCallum et al. (2005). We call the FSM state set described above the *positive* state set ($\mathcal{S}_1$), and duplicate the exact same set of states, and call the new set *negative* state set ($\mathcal{S}_0$). We then add a starting state(S_s), and add non-deterministic transitions from S_s to every state in $\mathcal{S}_1$. We then add the same transitions for $\mathcal{S}_0$. We now arrive at a new FSM structure where upon arriving at the starting state, one makes a non-deterministic decision to enter either the positive set or the negative set and stay in that set until reaching the end of the edit sequence, since no transitions are allowed across the positive and negative set. Each edit operation sequence can now be associated with a sequence of positive states as well as a sequence of negative states. The intuitive idea is that during training, we want to maximize the weights of the positive examples in the positive state set and minimize their weights in the negative state set, and vice versa. In other words, we want the positive state set to attract positive examples but push away negative examples. Figure 1 illustrates two example valid edit sequences in the FSM, one in the positive state set and one in the negative state set.

Formally, the partition function Z in (1) is defined as the sum of weights of all valid edit sequences in both the positive set and negative set. Features extracted from positive states are disjoint from features extracted from negative states.

$$Z = \sum_{\mathbf{e}: \mathbf{e}.\mathbf{a} \subseteq S_s + \{\mathcal{S}_0 \bigcup \mathcal{S}_1\}^*} \prod_{i=1}^{|\mathbf{e}|} \exp \theta \cdot \mathbf{f}(e_{i-1}, e_i, \tau_\mathbf{t}, \tau_\mathbf{h})$$

Recall $z \in \{0, 1\}$ is the binary judgment indicator variable. The conditional probability of z is obtained by marginalizing over all edit sequences that have state transitions in the state set corresponding to z:

$$P(z \mid \tau_\mathbf{t}, \tau_\mathbf{h}) = \sum_{\mathbf{e}: \mathbf{e}.\mathbf{a} \subseteq S_s + \mathcal{S}_z^*} P(\mathbf{e} \mid \tau_\mathbf{t}, \tau_\mathbf{h}) \quad (2)$$

The L_2-norm penalized log-likelihood over n training examples ($\mathcal{L}$) is our training objective function:

$$\mathcal{L} = \sum_{j=1}^{n} \log(P(z^{(j)} \mid \tau_\mathbf{t}^{(j)}, \tau_\mathbf{h}^{(j)})) - \frac{\|\theta\|^2}{2\sigma^2} \quad (3)$$

At test time, the z with higher probability is taken as our prediction outcome.

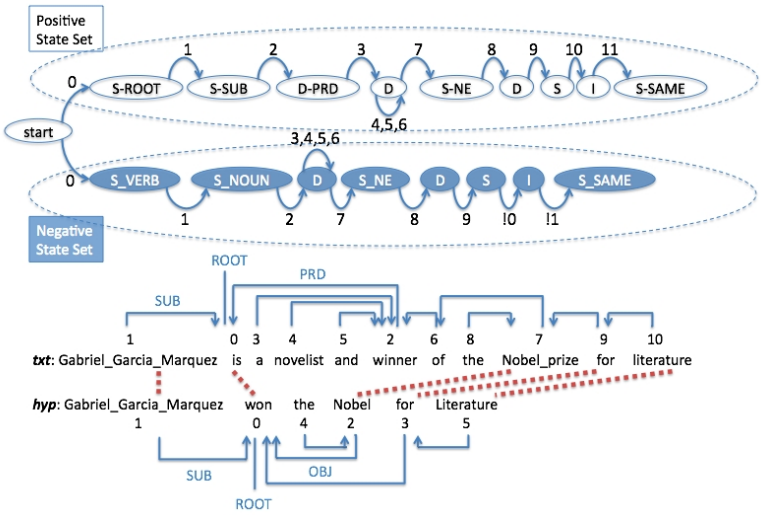

Figure 1: This diagram illustrates the FSM architecture. There is a single start state, and we can transit into either the positive state set (nodes that are not shaded), or the negative state set (shaded nodes). Here we show two examples of valid edit sequences. They result in the same alignment structure as show in the bottom half of the diagram (dotted lines across the two sentences are alignment links). Numbers over the arcs in the state diagram denote the edit sequence index, and numbers under each word in the parse tree diagram denote each node's level-order index number.

3 Parameter Estimation

We used Expectation Maximization method since the objective function given in (3) is non-convex. In the M-step, finding the optimal parameters under the current model expectation involves computing forward-backward style dynamic programming (DP) in a three-dimensional table (two for inputs and one for states) and optimization using L-BFGS method. In practice the resulting DP table can be quite large (for a sentence pair of length 100, and 2 sets of 45 states, we obtain 900,000 entries). We improved efficiency by pruning out partial sequences that do not lead to a complete valid sequence and pre-compute the state-transition table and features.

4 Edit Operations

Table 1 lists the groups of edit operations we designed and their descriptions. Not shown in the table are three default edits (*insert*, *delete* and *substitute*), which fire when none of the more specific edit operations match. Edit operations listed in the the top-left section capture basic matching, deletion and insertion of surface text, part-of-speech tags and named-entity tags. The top-right section capture alignments of semantically related words, based on relational information extracted from various linguistic resources, such as WordNet and NomBank. And the bottom section capture syntactic edits. Note that multiple edit operations can fire at the same edit position if conditions are matched (e.g., we can choose to delete if there are more words to edit in *txt*, or to insert if there are more words to edit in *hyp*).

5 Features

One of the most distinctive advantages of our model compared to previous tree-edit based models is the ability to include a wide range of non-independent, rich linguistic features. The features we employed can be broken down into two categories. The first category is *zero-order* features that model the current edit step. They consist of a conditioning property of the current edit, and the current state in the FSM. The second category is *first-order* features that capture state transitions, by concatenating the current FSM state with the previous FSM state. One simple form of zero-order feature is the current FSM state itself. The FSM states already carry a lot of information about the current edits. Conditioning properties are used to further describe the current edit. They are often more fine-grained and complex (e.g.,

Surface edits		Semantic edits	
{I,D,S}-{POS}	insert/delete/substitute words of a POS type, where POS is *noun*, *verb* or *proper noun*	S-SYNONYM	substitute two words that are synonyms
		S-HYPERNYM	substitute two words that are hypernyms
{I,D,S}-NE	insert/delete/substitute named-entity words	S-ANTONYM	substitute two words that are antonyms
{I,D,S}-LIKE	insert/delete/substitute words that expresses likelihood, e.g., *maybe*, *possibly*	S-ACRONYM	substitute two words in which one is an acronym of the other
{I,D,S}-MODAL	insert/delete/substitute modal verbs, e.g., *can*, *could*, *may*	S-NOMBANK	substitute two words that are related according to NomBank
S-{SAME/DIFF}	the words being substituted are the same or different	S-NUM-0,1	substitute two words that are both numerical values, and 1 if they match, 0 if they mismatch
Syntactic edits			
{I,D,S}-ROOT	insert/delete/substitute root of the trees		
{I,D,S}-{REL}	insert/delete/substitute a tree node of grammatical relation type, where REL is either SUB, OBJ, VC or PRD		

Table 1: List of edit operations. I for INSERT, D for DELETE, and S for SUBSTITUTE.

syntactic-matching conditions listed below). To give an example, in Figure 1, the second edit operation in the example sequence is S-NE. A matching condition feature that fires with this state could be *substitute_NE_type_PERSON*, which tells us exactly what type of named-entity is being substituted.

It is notable that in designing edit operations and features, there is a continuum of choice in terms of how much information to be encoded as features versus edit operations. To better illustrate the trade-off, consider the two extreme cases of this continuum. At one extreme, we can design a system where there are only three basic edit operations, and all extra information in our current set of edit operations can be encoded as features. For example, in this case edit operation S-NE would become S with feature *substitute_NE*. The other extreme is to encode every zero-order feature as a separate edit operation. The amount of information encoded in the zero-order features and edit operations is the same in both cases, but the difference lies in first-order features and efficiency. When encoding more information as edit operations (and thus more states in FSM), first-order features become much more expressive; whereas when encoding more information as features, computation becomes cheaper as the number of possible state transition sequences is reduced. In our experiments, we aim to keep a minimal set of edit operations that are meaningful but not overly verbose, and encode additional information as features. Each feature is a binary feature initialized with weight 0.

Due to space limitation, we list the most important zero-order features. Many of these features are inspired by MacCartney et al. (2006) and Snow et al. (2006), but not as sophisticated.

Word matching features. These features detect if a text word and a hypothesis word match the following conditions:

1. have the same lemma
2. one is a phrase and contains the other word
3. are multi-word phrases and parts match
4. have the same/different named-entity type(s) + the named-entity type(s)

Tree structure features. These features try to capture syntactic matching/mismatching information from the labeled dependency parse trees. 1. whether the roots of the two trees are aligned
2. parent-child pair match
3. (2.) and labels also match
4. (2.) and labels mismatch
5. (4.) and detailing the mismatching labels
6. parent+label match, child mismatch
7. child and label match, parents are {hyper/syno/anto}nym
8. looking for specific SUB/OBJ/PRD construct as in Snow et al. (2006).

6 Preprocessing

In all of our experiments, each input pair of text and hypothesis sentence is preprocessed as following: Sentences were first tokenized by the standard Penn TreeBank tokenization script, and then we used MXPOST tagger (Ratnaparkhi, 1996) for part-of-speech (POS) tagging. POS tagged sentences were then parsed by MST-Parser (McDonald et al., 2005) to produce labeled dependency parse trees. The parser was trained

on the entire Penn TreeBank. The last step in the pipeline is named-entity tagging using Stanford NER Tagger (Finkel et al., 2005).

7 RTE Experiments

Given an input text sentence and a hypothesis sentence, the task of RTE is to make predictions about whether or not the hypothesis can be entailed from the text sentence. We use standard evaluation datasets RTE1-3 from the Pascal RTE Challenges (Dagan et al., 2006). For each RTE dataset, we train a tree-edit CRF model on the training portion and evaluate on the testing portion. We report accuracy of classification results, and precision and recall for the true entailment class. There is a balanced positive-negative sample distribution in each dataset, so a random baseline gives 50% classification accuracy. We used RTE1 for feature selection and tuning σ in the L_2 regularizer ($\sigma = 5$ was used). RTE2 and RTE3 were reserved for testing.

Our system is compared with four systems on RTE2 and three other systems on the RTE3 dataset.[1] We chose these systems for comparison because they make use of syntactic dependencies and lexical semantic information. Notably other systems that give state-of-the-art performance on RTE use non-comparable techniques such as theorem-proving and logical induction, and often involve significant manual engineering specifically for RTE, thus do not make meaningful comparison to our model.

For RTE2, Kouylekov and Magnini (2006) experimented with various TED cost functions and found a combination scheme to work the best for RTE. Vanderwende et al. (2006) used syntactic heuristic matching rules with a lexical-similarity back-off model. Nielsen et al. (2006) extracted features from dependency path, and combined them with word-alignment features in a mixture of experts classifier. Zanzotto et al. (2006) proposed a syntactic cross-pair similarity measure for RTE.

For RTE3, Harmeling (2007) took a similar classification-based approach with transformation sequence features. Marsi et al. (2007) described a system using dependency-based paraphrasing techniques for RTE. de Marneffe et al. (2006) described a system where best alignments between the sentence pairs were first found, then classification decisions were made based on these alignments.

[1] Different systems are used for comparison because none of these systems reported performance on both datasets.

RTE2	**Acc.%**	**Prec.%**	**Rec.%**
Vanderwende et al., 2006	60.2	59.0	67.0
K&M, 2006	60.5	58.9	70.0
Nielsen et al., 2006	61.1	59.0	73.3
Zanzotto et al., 2006	63.9	60.8	78.0
Tree-edit CRF	63.0	61.7	68.5
RTE3	**Acc.%**	**Prec.%**	**Rec.%**
Marsi et al., 2007	59.1	-	-
Harmeling, 2007	59.5	-	-
de Marneffe et al., 2006	60.5	61.8	60.2
Tree-edit CRF	61.1	61.3	65.3

Table 2: Results on RTE2 and RTE3 dataset. Results for de Marneffe et al. (2006) were reported by MacCartney and Manning (2008).

Table 2 presents RTE results. Our model performs competitively on both datasets. On RTE2, our model gives second best performance among the methods we compare against, and the difference in accuracy from the best system is quite small (7 out of 800 examples). We observe a larger gap in recall, suggesting our method tends to give higher precision, which is also commonly found in other syntax-based systems (Snow et al., 2006). It is worth noting that Zanzotto et al. (2006) achieved second place in the official RTE2 evaluation. On RTE3, our model outperforms the other syntax-based systems compared. In particular, out system gives the same precision level as the second best system (de Marneffe et al., 2006) without sacrificing as much recall, which is the most common drawback found in syntax-based systems.

8 QA Experiments

A second Tree-edit CRF model was trained for the task of answer selection for Question Answering. In this task, the input pair consists of a short factoid question (e.g., *Who beat Floyd Patterson to take the title away?*) and an answer candidate sentence (e.g., *He saw Ingemar Johansson knock down Floyd Patterson seven times there in winning the heavyweight title.*). The pair is judged positive if the answer candidate sentence correctly answers the question and provides sufficient con-

System	MAP	MRR
Punyakanok et al., 2004	0.4189	0.4939
Cui et al., 2005	0.4350	0.5569
Wang et al., 2007	0.6029	0.6852
H&S, 2010	**0.6091**	0.6917
Tree-edit CRF	0.5951	**0.6951**

Table 3: Results on QA task reported in Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

textual support (i.e., does not merely contain the answer key, for example, *"Ingemar Johansson was a world heavyweight champion"* would not be a correct answer). We followed the same experimental setup as Wang et al. (2007) and Heilman and Smith (2010). The training portion of the dataset consists of 5919 manually judged Q/A pairs from previous QA tracks at Text REtrieval Conference (TREC 8–12). There are also 1374 Q/A pairs for development and 1866 Q/A pairs for testing, both from the TREC 3 evaluation. The task is framed as a sentence retrieval task, and thus Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are reported for the ranked list of most probable answer candidates. We compare out model with four other systems. Wang et al. (2007) proposed a Quasi-synchronous Grammar formulation of the problem which also models alignment as structured latent variables, but in a generative probabilistic model. Their method gives the current state-of-the-art performance on this task. Heilman and Smith (2010) presented a classification-based approach with tree-edit features extracted from a tree kernel. Cui et al. (2005) developed a dependency-tree based information discrepancy measure. Punyakanok et al. (2004) used a generalized Tree-edit Distance method to score mappings between dependency parse trees. All systems were evaluated against the same dataset as the one we used. Results of replicated systems for the last two were reported by Wang et al. (2007), with lexical-semantic augmentation from WordNet.

Results in Table 3 show that our model gives the same level of performance as Wang et al. (2007), with no statistically significant difference ($p > 5$ in sign test). Both systems out-perform the other two earlier systems significantly.

9 Discussion

Our experiments on RTE and QA applications demonstrated that Tree-edit CRF models provide results competitive with previous syntax-based methods. Even though the improvements were quite moderate in some cases, the important point is that our model provides a novel principled framework. It works across different problem domains with minimal domain knowledge and feature engineering, whereas previous methods are only engineered for a particular task and are hard to generalize to new problems.

While the current Tree-edit CRF model can model a large set of linguistic phenomenon and tree-transformations, it has some clear limitations. One of the biggest drawbacks is the lack of support for modeling phrasal re-ordering, which is a very common and important linguistic phenomena. It is not straightforward to implement reordering in the current model because it breaks the word-order constraint which admits tractable forward-backward style dynamic programming. However, this shortcoming can be addressed partially by extending the model to deal with constrained re-ordering per Zhang (1996).

10 Related Work

Tree Edit Distance (TED) have been studied extensively in theoretical and algorithmic research (Klein, 1989; Zhang and Shasha, 1989; Bille, 2005). In recent years we have seen many work on applying TED based methods for NLP-related tasks (Punyakanok et al., 2004; Kouylekov and Magnini, 2006; Harmeling, 2007; Mehdad, 2009). Mehdad (2009) proposed a method based on particle swarm optimization technique to automatically learn the TED cost function. Another work that also developed an interesting approach to stochastic tree edit distance is Bernard et al. (2008), but unfortunately experiments in the paper were limited to digit recognition and tasks on small artificial datasets.

Many different approaches to modeling sentence alignment have been proposed before (Haghighi et al., 2005; MacCartney et al., 2008). Haghighi et al. (2005) treated alignment finding in RTE as a graph matching problem

between sentence parse trees. MacCartney et al. (2008) described a phrase-based alignment model for MT, trained by the Perceptron learning algorithm. A line of work that offers similar treatment of alignment to our model is the Quasi-synchronous Grammar (QG) (Smith and Eisner, 2006; Wang et al., 2007; Das and Smith, 2009). QG models alignments between two parse trees as structured latent variables. The generative story of QG describes one that builds the parse tree of one sentence, loosely conditioned on the parse tree of the other sentence. This formalism prefers but is not confined to tree isomorphism, therefore possesses more model flexibility than synchronous grammars.

The work of McCallum et al. (2005) inspired the discriminative training framework that we used in our experiments. They presented a String Edit Distance model that also learns alignments as hidden structures for simple tasks such as restaurant name matching.

Our work is also closely related to other recent work on learning probabilistic models involving structural latent variables (Clark and Curran, 2004; Petrov et al., 2007; Blunsom et al., 2008; Chang et al., 2010). The Tree-edit CRF model we present here is a new addition to this family of interesting models for discriminative learning with structural latent variables.

11 Conclusion

We described a Tree-edit CRF model for predicting semantic relatedness of pairs of sentences. Our approach generalizes TED in a principled probabilistic model that embeds alignments as structured latent variables. We demonstrate a wide-range of lexical-semantic and syntactic features can be easily incorporated into the model. Discriminatively trained, the Tree-edit CRF led to competitive performance on the task of Recognizing Textual Entailment and answer selection for Question Answering.

References

Bernard, M., L. Boyer, A. Habrard, and M. Sebban. 2008. Learning probabilistic models of tree edit distance. *Pattern Recognition*, 41(8):2611–2629.

Bille, P. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239.

Blunsom, P., T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-HLT*.

Chang, Ming-Wei, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of NAACL-HLT*.

Clark, S. and J. R. Curran. 2004. Parsing the wsj using ccg and log-linear models. In *Proceedings of ACL*.

Cui, Hang, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR*.

Dagan, I., O. Glickman, and B. Magnini. 2006. The pascal recognising textual entailment challenge. *Machine Learning Challenges, LNCS*, 3944:177–190.

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

de Marneffe, M.-C., B. MacCartney, T. Grenager, D. Cer, A. Rafferty, and C. D. Manning. 2006. Learning to distinguish valid textual entailments. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

de Salvo Braz, R., R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment and question-answering. In *Proceedings of AAAI*.

Finkel, J. R., T. Grenager, and C. D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*.

Finkel, J. R., C. D. Manning, and A. Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of EMNLP*.

Haghighi, A., A. Y. Ng, and C. D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP*.

Harmeling, S. 2007. An extensible probabilistic transformation-based approach to the third recognizing textual entailment challenge. In *Proceedings of ACL PASCAL Workshop on Textual Entailment and Paraphrasing*.

Heilman, M. and N. A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL-HLT*.

Jijkoun, V. and M. de Rijke. 2005a. Recognizing textual entailment: Is word similarity enough?. In *Machine Learning Challenge Workshop*, volume 3944 of *LNCS*, pages 449–460. Springer.

Jijkoun, V. and M. de Rijke. 2005b. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on RTE*.

Klein, P. N. 1989. Computing the edit-distance between unrooted ordered trees. In *Proceedings of European Symposium on Algorithms*.

Kouylekov, M. and B. Magnini. 2006. Tree edit distance for recognizing textual entailment: Estimating the cost of insertion. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

MacCartney, Bill and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of Workshop on Textual Entailment and Paraphrasing at ACL 2007*.

MacCartney, B. and C. D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of COLING*.

MacCartney, B., T. Grenager, M.-C. de Marneffe, D. Cer, and C. D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of HLT-NAACL*.

MacCartney, B., M. Galley, and C. D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*.

Marsi, E., E. Krahmer, and W. Bosma. 2007. Dependency-based paraphrasing for recognizing textual entailment. In *Proceedings of ACL PASCAL Workshop on Textual Entailment and Paraphrasing*.

McCallum, A., K. Bellare, and F. Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of UAI*.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Mehdad, Yashar. 2009. Automatic cost estimation for tree edit distance using particle swarm optimization. In *Proceedings of ACL*.

Moldovan, D., C. Clark, S. Harabagiu, and S. Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL*.

Nielsen, R. D., W. Ward, and J. H. Martin. 2006. Toward dependency path based entailment. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Papineni, K., S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Petrov, S., A. Pauls, and D. Klein. 2007. Discriminative log-linear grammars with latent variables. In *Proceedings of NIPS*.

Punyakanok, V., D. Roth, and W. Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of AI-Math*.

Raina, R., A. Y. Ng, , and C. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI*.

Ratnaparkhi, Adwait. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP*.

Smith, D. A. and J. Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*.

Snow, R., L. Vanderwende, and A. Menezes. 2006. Effectively using syntax for recognizing false entailment. In *Proceedings of HLT-NAACL*.

Vanderwende, L., A. Menezes, and R. Snow. 2006. Microsoft research at rte-2: Syntactic contributions in the entailment task: an implementation. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Wang, M., N. A. Smith, and T. Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for question answering. In *Proceedings of EMNLP-CoNLL*.

Zanzotto, F. M., A. Moschitti, M. Pennacchiotti, and M.T. Pazienza. 2006. Learning textual entailment from examples. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Zhang, K. and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18.

Zhang, K. 1996. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3):205–222.