Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions: Appendix with Experimental Details and Errata

Michael Heilman Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{mheilman,nasmith}@cs.cmu.edu

1 Introduction

This document provides additional details about the experiments described in (Heilman and Smith, 2010). Note that while this document provides information about the datasets and experimental methods, it does not provide further results.

If you have any further questions, please feel free to contact the first author. The preprocessed datasets (i.e., tagged and parsed) will be made available for research purposes upon request.

2 Experiments

Experiments were conducted to evaluate tree edit models for three tasks: recognizing textual entailment (Giampiccolo et al., 2007), paraphrase identification (Dolan et al., 2004), and an answer selection task (Wang et al., 2007) for question answering (Voorhees, 2004).

The feature set and first tree edit model were developed for the paraphrase task, using cross-validation on the training data, and then applied to the other tasks with very few modifications and no further tuning.

All datasets were POS-tagged using Ratnaparkhi's (1996) tagger and parsed for dependencies using the MST Parser (McDonald et al., 2005).

Features were computed from POS and edge label information in the dependency parses. Each node in the dependency trees, rather than having a single label, contained multiple pieces of information: a word lemma, a part of speech (e.g., "PRP" for pronouns, "NNP" for proper nouns), a pointer to its parent (i.e., the edge), and a label for the edge from itself to its parent (e.g., "SUB" for nouns that act as subjects to their parents, "OBJ" for nouns that act as objects, etc.). A similar approach is taken in (Culotta and Sorensen, 2004).

The WordNet API (Miller et al., 1990) was used for lemmatization—and only for that.

2.1 Recognizing Textual Entailment

A tree edit model was trained for recognizing textual entailment (RTE). Here, an instance consists of a "premise," which is a sentence or paragraph about a particular topic or event, and a "hypothesis," which is a single, usually short, sentence that may or may not follow from the premise. The task is to decide whether or not the hypothesis is entailed by the premise (Giampiccolo et al., 2007). The definition of entailment is informal rather than strict: a hypothesis is labeled as entailed if a reasonable person, upon reading the premise, would believe that the hypothesis is necessarily true. For example, a hypothesis *Fred was out of money* would be entailed by the premise *Fred spent his last dime at the book store* (even though technically he might have a coin other than a dime).

Tree edit sequences were extracted in one direction, from premise to hypothesis. We could have modeled transformations in both directions, but felt that it is counter-intuitive to model *adding* information through extensive insertions, for both entailment and answer selection.

Since premises may consist of multiple sentences, we attach sentences as children of dummy root nodes, for both the premise and hypothesis. For example, a tree for a premise with two sentences would have a single dummy root node with three root nodes as children, in the same order as the original sentence. A hypothesis would also have a dummy root node, with the single hypothesis sentence as a child. Note that a valid edit sequence would necessarily contain edits for deleting the extra root nodes in the premise.

The model was trained on the development set

(i.e., training data) for RTE-3 along with all the data from the RTE-1 and RTE-2 tasks. Similar training procedures It was then evaluated on the RTE-3 test set. We report precision and recall for true entailments, and overall accuracy. Precision is the percentage of predicted entailments that were labeled true entailments in the gold standard. Recall is the percentage of true entailments that were predicted to be entailments. Overall accuracy is the percentage of predictions that matched the gold standard (either "entailed" or "not entailed").

We compare to four systems that use syntactic dependencies and lexical semantic information. The top-performing RTE systems often involve significant manual engineering for the RTE task. Also, many employ techniques that make them not very comparable to our approach (e.g., theorem proving). We also note that Kouylekov and Magnini (2005) report 55% accuracy for RTE-2 using TED, but they do not report results for RTE-3. See Giampiccolo et al. (2007) for more RTE-3 results.

De Marneffe et al. (2006) described an RTE system that finds word alignments and then classifies sentence pairs based on those alignments. MacCartney and Manning (2008) used an inference procedure based on Natural Logic, leading to a relatively high-precision, low-recall system. MacCartney and Manning (2008) also tested a hybrid of the natural logic system and the complementary system of de Marneffe et al. (2006) to improve coverage.

Harmeling (2007) took an approach similar to ours involving classification based on transformation sequences, but with less general operations and a more complex, heuristic procedure for finding sequences.

The systems described above, as well as our method, separate the steps of alignment and classification. First, the syntactic tree for the premise is aligned to the tree for the hypothesis; then, various features of this alignment are used to classify the sentence pair. In contrast, other work, such as (), mark hypotheses as entailed if their level of alignment with the premise exceeds some threshold, effectively deciding textual entailment in a single step. The analog in terms of tree edits to this one step alignment-based entailment decision would be to predict entailment based solely on the number of edits required. We initially tried such an approach, but it did not perform much above random chance.

2.2 Paraphrase Identification

A tree edit model was trained and tested for paraphrase identification using the the Microsoft Research Paraphrase Corpus (Dolan et al., 2004). The task is to identify whether two sentences convey essentially the same meaning. As with textual entailment, the notion of paraphrase is informal: paraphrases need not convey *exactly* the same meaning.

The standard training set was used to train the tree edit classification model to distinguish between true and false paraphrases. Since there is no predefined direction for paraphrase pairs, we extracted two sequences for each pair (one in each direction) and summed the feature values. The model was evaluated with the standard test set.

We evaluated the model in terms of accuracy, positive class precision (i.e., % of predicted positive paraphrases that had positive gold-standard labels), and positive class recall (i.e., % of positive gold-standard labels that were predicted to be positive paraphrases).

We compare to two of the best performance approaches to paraphrase. One approach, by Wan et al. (2006), uses an SVM classifier with features based on syntactic dependencies, TED, unigram overlap, and BLEU scores (Papineni et al., 2002). The other system, by Das and Smith (2009), is based on a quasi-synchronous grammar (QG; Smith and Eisner, 2006), a probabilistic model that allows loose alignments between trees but prefers tree isomorphism. In addition to syntactic dependencies, the QG model utilizes entity labels from BBN Identifinder (Bikel et al., 1999) and lexical semantics knowledge from WordNet. Das and Smith (2009) also use a product of experts (PoE) (Hinton, 1999) to combine the QG model with lexical overlap features.

2.3 Answer Selection for Question Answering

A tree edit model was trained for answer selection in question answering (QA). In this task, an instance consists of a short factual question (e.g., *Who wrote the 'Tale of Genji'?*) and a candidate answer sentence retrieved by the information retrieval component of a question answering system. For a positive instance, the text will correctly answer the question—though perhaps indirectly. It may also

contain various extraneous information (e.g., *Kano script made possible the development of a secular Japanese literature, beginning with such Late Heian classics as Lady Murasaki's "Tales of Genji."*). For a given set of questions, the task here is to *rank* candidate answers (Wang et al., 2007).

The experimental setup is the same as in Wang et al. (2007). In fact, we used the same preprocessed data as used in that paper. The model was trained and tested on data from previous QA tracks at the Text REtrieval Conference (TREC-8 through TREC-12). The original TREC data provides question and answer patterns along with pools of documents returned by participating teams for each question.

To generate the candidate answer sets for their answer selection experiments, Wang et al. (2007) automatically selected sentences from each questions' document pool that contained one or more non-stopwords that appeared in the question. Manual judgments were produced for all of the TREC-13 data, and for the first 100 questions from TREC 8-12.

The output for this answer selection task is slightly different from the output in TREC QA evaluations, where answers are short strings (e.g., *1955*) paired with document IDs (Wang et al., 2007). Here, answers are complete sentences without document IDs, and they are correct only if the sentence contains a correct answer (e.g., *1955*) and also addresses the question (e.g., rather than referring to *1955* in some other context). Wang et al. (2007) exclude from the test set those questions that have only correct answers or only incorrect answers, as well as answer candidates longer than 40 words.

We trained the tree edit model on the manually judged positive and negative QA pairs extracted by Wang et al. (2007) from TREC 8-12. Since the goal of the task is to rank answer candidates rather than classify them, we rank the answer candidates for a given question by their estimated probabilities of correctness according to a trained logistic regression classifier.

We tested our model with QA pairs from TREC-13.[1] We report Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR), which are information retrieval measures for ranked lists.

MAP provides a single measure of the quality of the ranked answers that averages across various levels of recall (i.e., how many of the gold standard correct answers are included in the ranking up to a certain point). Following Manning et al. (2008), let Q be the set of questions (queries). Let m_j be the number of gold standard correct answers (relevant documents) for the j^{th} question, and let R_{jk} be the list of ranked answers from the top-ranked answer down to and including the the k^{th} correct answer. Also, let $Prec(R)$ be the precision for a given list of answers. Then, MAP is defined as follows:

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Prec(R_{jk}). \quad (1)$$

MRR is a metric that focuses more on the top-ranked results. It is the average of the the multiplicate inverse of the rank of the first gold standard correct answer. If f_j is the rank of the the first correct answer in the ranked answers for the j^{th} question, then

$$MRR = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{r_j}. \quad (2)$$

Tree edit sequences were extracted only in one direction, from answer to question. As in the RTE task, modeling edit sequences from short questions to long answers is counter-intuitive since the answers contain extra information.

We compare our tree edit model to three other systems as they are reported by Wang et al. (2007). Wang et al. use a QG model, incorporating information from dependency trees, entity labels from BBN Identifinder (Bikel et al., 1999), and lexical semantics knowledge from WordNet (Miller et al., 1990). Cui et al. (2005) developed an information theoretic measure based on dependency trees. Punyakanok et al. (2004) used a generalization of TED to model the QA pairs. For their experiments, Wang et al. also extended both of the latter models to utilize WordNet.

[1] Wang et al. (2007) set aside 84 questions as a development set. We did not utilize these data for training, development, or testing.

3 Errata

The section "3.2 Tree Kernel Heuristic" of the original paper contains mistakes pertaining to the use of the penalty term $lamba$. [2] We described this term as a penalty on the length of subsequences. However, for such a penalty to occur, $\lambda^{2|J_1|}$ would need to appear between the summation and product operators in Equation 4 of the original paper, as in moschitti-06.

It appears that in our experiments, $\lambda^2 = 0.0625$ simply served to make sure that the node similarity function in Equation 2 of the original paper, which in the original paper could range from 0 to 4, would not exceed 1.0. In the fixed version of the kernel we present here, we normalize the node similarity function to a zero to one range (by multiplying by 0.25). We also include the $\lambda^{2|J_1|}$ term but simply set it to 1.0. These modifications do not appear to substantially affect the search process or the performance of the full system. E.g., for paraphrase identification, using the modified version of the kernel led to 73.3% accuracy, 76.2% positive class precision, and 87.0% positive class recall. The original paper reported 73.2% accuracy, 75.7% precision, and 87.8% recall.[3]

The following section provides a revised description of the tree kernel. The only substantive modifications are in Equation 3 and Equation 5. (Note that what we implemented for our original experiments corresponds to the description in the original paper).

3.1 Revised Excerpt on the Tree Kernel Search Heuristic

...

To define our kernel, we begin with a similarity function for pairs of nodes n_1 and n_2 that depends on their lemmas, POS tags, edge labels, and sides with respect to their parents:[4]

$$s(n_1, n_2) = \delta(l(n_1), l(n_2)) \times \sum_{f \in \{l,e,p,s\}} 0.25 * \delta(f(n_1), f(n_2)) \quad (3)$$

where δ returns 1 if its arguments are equivalent, 0 otherwise. l, e, p, and s are used here as functions to select the lemma, edge label, POS, and side of a node. Equation 3 encodes the linguistic intuition that the primary indicator of node similarity should be a lexical match between lemmas. If the lemmas match, then edge labels, POS, and the locations (sides) relative to their parents are also considered. The 0.25 multiplier normalizes the function to a 0 to 1 range.

The kernel is defined recursively (starting from the roots), where n_i is a node in the set of nodes N_{T_i} in tree T_i:

$$K(T_1, T_2) = \sum_{n_1 \in \{N_{T_1}\}} \sum_{n_2 \in \{N_{T_2}\}} \Delta(n_1, n_2) \quad (4)$$

$$\Delta(n_1, n_2) = \mu \left(\lambda^2 s(n_1, n_2) + \sum_{J_1, J_2, |J_1| = |J_2|} \lambda^{2|J_1|} \prod_{i=1}^{|J_1|} \Delta(c_{n_1}[J_{1i}], c_{n_2}[J_{2i}]) \right) \quad (5)$$

$J_1 = \langle J_{11}, J_{12}, J_{13}, \ldots \rangle$ is an index sequence associated with any *contiguous* ordered sequence of children c_{n_1} of node n_1 (likewise for J_2). J_{1i} and J_{2i} point to the ith children in the two sequences. $|\cdot|$ returns the length of a sequence.

The kernel includes decay factor μ for the height of the subtree, as in Collins and Duffy (2001) and Moschitti (2006); and λ for the length of subsequences, as in Moschitti (2006).

The main difference between our kernel and the CTK is that we sum over all pairs of subtrees (Equation 4). In contrast, the CTK only considers only one pair of subtrees. When the CTK is applied to relation extraction by Culotta and Sorensen (2004), each subtree is the smallest common subtree that includes the entities between which

[2] Thanks to Yehoshua Gev for pointing out that the tree kernel equation was incorrect.

[3] As in the original paper, μ was set to 0.25 to encourage the search to consider edits leading to smaller matches (e.g., of individual parent-child dependencies) before larger ones.

[4] The side of a node relative to its parent in a dependency tree is important: two parent nodes with the same children should not be considered exact matches if children are on different sides (e.g., *defeated the insurgents* and *the insurgents defeated*).

a relation may exist (e.g., the subtree for *Texas-based energy company Exxon Mobil* when extracting ORGANIZATION-LOCATION relations).

References

D. M. Bikel, R. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34.

M. Collins and N. Duffy. 2001. Convolution kernels for natural language. In *Proc. of NIPS*.

H. Cui, R. Sun, K. Li, M. Kan, , and T. Chua. 2005. Question answering passage retrieval using dependency relations. In *Proc. of ACM-SIGIR*.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of ACL*.

D. Das and N. A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

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

B. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proc. of COLING*.

D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan, editors. 2007. *The third pascal recognizing textual entailment challenge*.

S. Harmeling. 2007. An extensible probabilistic transformation-based approach to the third Recognizing Textual Entailment challenge. In *Proc. of ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

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

G. E. Hinton. 1999. Product of experts. In *Proc. of ICANN*.

M. Kouylekov and B. Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proc. of the PASCAL RTE Challenge*.

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

C. D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. 1990. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4).

A. Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proc. of ECML*.

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

V. Punyakanok, D. Roth, and W. Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proc. of the 8th International Symposium on Artificial Intelligence and Mathematics*.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proc. of EMNLP*.

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

E. M. Voorhees. 2004. Overview of TREC 2004. In *Proc. of TREC*.

S. Wan, M. Dras, R. Dale, and C. Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proc. of the Australasian Language Technology Workshop*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.