

Recognizing Textual Entailment with Tree Edit Distance Algorithms

Milen Kouylekov^{1,2} and Bernardo Magnini²

ITC-irst, Centro per la Ricerca Scientifica e Tecnologica ¹

University of Trento²

38050, Povo, Trento, Italy

milen@kouylekov.net, magnini@itc.it

Abstract

This paper summarizes ITC-irst participation in the PASCAL challenge on Recognizing Textual Entailment (RTE). Given a pair of texts (the *text* and the *hypothesis*), the core of the approach we present is a tree edit distance algorithm applied on the dependency trees of both the text and the hypothesis. If the distance (i.e. the cost of the editing operations) among the two trees is below a certain threshold, empirically estimated on the training data, then we assign an entailment relation between the two texts.

1 Introduction

The problem of language variability (i.e. the fact that the same information can be expressed with different words and syntactic constructs) has been attracting a lot of interest during the years and it poses significant issues in front of systems aimed at natural language understanding. The example below shows that recognizing the equivalence of the statements *came in power, was prime-minister* and *stepped in as prime-minister* is a challenging problem.

- Ivan Kostov came in power in 1997.
- Ivan Kostov was prime-minister of Bulgaria from 1997 to 2001.
- Ivan Kostov stepped in as prime-minister 6 months after the December 1996 riots in Bulgaria.

While the language variability problem is well known in Computational Linguistics, a general unifying framework has been proposed only recently in (Dagan and Glickman 2004). In this approach, language variability is addressed by defining the notion of *entailment* as a relation that holds between two language expressions (i.e. a text T and an hypothesis H) if the meaning of H as interpreted in the context of T, can be inferred from the meaning of T. The entailment relation is directional as the meaning of one expression can entail the meaning of the other, while the opposite may not.

For our participation in the Pascal RTE Challenge we designed a system based on the intuition that the probability of an entailment relation between T and H is related to the ability to show that the whole content of H can be mapped into the content of T. The more straightforward the mapping can be established, the more probable is the entailment relation. Since a mapping can be described as the sequence of editing operations needed to transform H into T, where each edit operation has a cost associated with it, we assign an entailment relation if the overall cost of the transformation is below a certain threshold, empirically estimated on the training data.

The paper is organized as follows. Section 2 presents the Tree Edit Distance algorithm we have adopted and its application to dependency trees. Section 3 describes the system which participated at the RTE challenge and in Section 4 we present and discuss the results we have obtained.

2 Tree Edit Distance on Dependency Trees

We adopted a tree edit distance algorithm applied to the syntactic representations (i.e. dependency trees) of both T and H. A similar use of tree edit distance has been presented by (Punyakanok et al. 2004) for a Question Answering system, showing that the technique outperforms a simple bag-of-word approach. While the cost function presented in (Punyakanok et al. 2004) is quite simple, for the RTE challenge we tried to elaborate more complex and task specific measures.

According to our approach, T entails H if there exists a sequence of transformations applied to T such that we can obtain H with an overall cost below a certain threshold. The underlying assumption is that pairs between which an entailment relation holds have a low cost of transformation. The kind of transformations we can apply (i.e. deletion, insertion and substitution) are determined by a set of predefined entailment rules, which also determine a cost for each editing operation.

We have implemented the tree edit distance algorithm described in (Zhang and Shasha 1990) and applied to the dependency trees derived from T and H. Edit operations are defined at the level of single nodes of the dependency tree (i.e. transformations on subtrees are not allowed in the current implementation). Since the (Zhang and Shasha 1990) algorithm does not consider labels on edges, while dependency trees provide them, each dependency relation R from a node A to a node B has been re-written as a complex label B-R concatenating the name of the destination node and the name of the relation. All nodes except the root of the tree are relabeled in such way. The algorithm is directional: we aim to find the better (i.e. less costly) sequence of edit operation that transform T (the source) into H (the target). According to the constraints described above, the following transformations are allowed:

- **Insertion:** insert a node from the dependency tree of H into the dependency tree of T. When a node is inserted it is attached with the dependency relation of the source label.
- **Deletion:** delete a node N from the dependency tree of T. When N is deleted all its children are attached to the parent of N. It is not required to

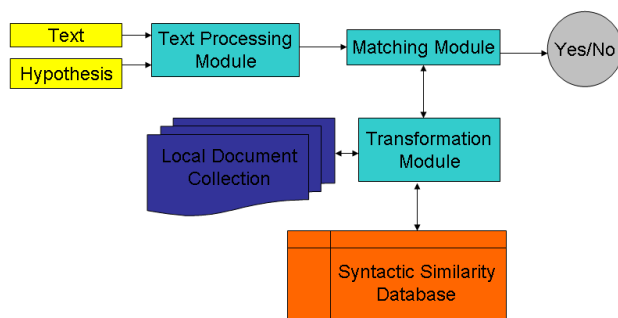


Figure 1: System architecture

explicitly delete the children of N as they are going to be either deleted or substituted on a following step.

- **Substitution:** change the label of a node N1 in the source tree into a label of a node N2 of the target tree. Substitution is allowed only if the two nodes share the same part-of-speech. In case of substitution the relation attached to the substituted node is changed with the relation of the new node.

3 System Architecture

The system is composed by the following modules, showed in Figure 1: (i) a text processing module, for the preprocessing of the input T/H pair; (ii) a matching module, which performs the mapping between T and H; (iii) a cost module, which computes the costs of the edit operations.

3.1 Text processing module

The *text processing module* creates a syntactic representation of a T/H pair and relies on a sentence splitter and a syntactic parser. For sentence splitting we used the Maximum entropy sentence splitter *MXTerm* (Ratnaparkhi 1996). For parsing we used *Minipar*, a principle-based English parser (Lin 1998) which has high processing speed and good precision.

3.2 Matching module

The *matching module* finds the best sequence of edit operations between the dependency trees obtained from T and H. It implements the edit distance algorithm described in Section 2. The module makes

requests to the *cost module* to receive the cost of the edit operations needed to transform T into H.

3.3 Cost module

The *cost module* returns the cost of an edit operation between tree nodes. To estimate such cost, we define a weight of each single word representing its relevance through the *inverse document frequency* (*idf*), a measure commonly used in *Information Retrieval*. If N is the number of documents in a text collection and N_w is the number of documents of the collection that contain word w then the *idf* of this word is given by the formula:

$$idf(w) = \log \frac{N}{N_w} \quad (1)$$

The weight of the *insertion* operation is the *idf* of the inserted word. The most frequent words (e.g. stop words) have a zero cost of insertion. In the current version of the system we are still not able to implement a good model that estimates the cost of the deletion operation. In order not to punish pairs with short contents of T we set the cost of deletion to 0. To determine the cost of substitution we used a dependency based thesaurus available at <http://www.cs.ualberta.ca/~lindek/downloads.htm>. For each word, the thesaurus lists up to 200 most similar words and their similarities. The cost of a *substitution* is calculated by the following formula:

$$subs(w_1, w_2) = ins(w_2) * (1 - sim(w_1, w_2)) \quad (2)$$

where w_1 is the word from T that is being replaced by the word w_2 from H and $sim(w_1, w_2)$ is the similarity between w_1 and w_2 in the thesaurus multiplied by the similarity between the corresponding relations. The similarity between relations is stored in a database of relation similarities obtained by comparing dependency relations from a parsed local corpus. The similarities have values from 1 (very similar) to 0 (not similar). If there is no similarity, the cost of substitution is equal to the cost of inserting the word w_2 .

3.4 Global Entailment Score

The *entailment* score of a given pair is calculated in the following way:

$$score(T, H) = \frac{ed(T, H)}{ed(, H)} \quad (3)$$

where $ed(T, H)$ is the function that calculates the edit distance cost and $ed(, H)$ is the cost of inserting the entire tree H. A similar approach is presented in (Monz and de Rijke 2001), where the entailment score of two document d and d' is calculated by comparing the sum of the weights (*idf*) of the terms that appear in both documents to the sum of the weights of all terms in d' .

To define the threshold that separates the positive from the negative examples we used the training set provided by the task organizers.

4 Results and Discussion

Table 1 shows the results obtained by the system on the two runs we submitted. The first run used the edit-distance approach on all the subtasks, while the second run used the edit distance for the Comparable Documents (CD) subtask task and a linear sequence of words for the rest of the tasks. We decided to do the second run because we wanted to evaluate the real impact of using deep syntactic analysis. Results are slightly better for the first run both in the cws (0.60 against 0.58) and recall (0.64 against 0.50).

A relevant problem we encountered, affecting about 30% of the pairs, is that the parser represents in a different way occurrences of similar expressions, making harder to apply edit transformations. For instance, “Wal-Mart” and “Wal-Mart Stores inc.” have different trees, being “Mart” the governing node in the first case and the governed node in the second. The problem could be addressed by changing the order of the nodes in T which is however complex because it introduces changes in the tree edit-distance algorithm. Another solution, which we intend to explore in the future, is the integration of specialized tools and resources for handling named entities and acronyms. In addition, for about 20% of the pairs, the parser did not produce the right analysis either for T or for H.

Another drawback of the tree-edit distance approach is that it is not able to observe the whole tree, but only the subtree of the processed node. For example, the cost of the insertion of a subtree in H

run	measure	CD	IE	MT	QA	RC	PP	IR	Overall
1	accuracy	0.78	0.48	0.50	0.52	0.52	0.52	0.47	0.55
	cws	0.89	0.50	0.55	0.49	0.53	0.48	0.51	0.60
	precision								0.55
	recall								0.64
2	accuracy	0.78	0.53	0.49	0.48	0.54	0.48	0.47	0.55
	cws	0.89	0.53	0.53	0.42	0.58	0.43	0.50	0.58
	precision								0.56
	recall								0.50

Table 1: ITC-irst results at PASCAL-RTE

could be smaller if the same subtree is deleted from T in prior or later stage.

The current implementation of the system does not use resources (e.g. WordNet, paraphrases in (Lin and Pantel 2001), entailment patterns as acquired in (Szpektor et al. 2004)) that could significantly wide the application of entailment rules and, consequently, improve performances. We estimated that for about 40% of the the true positive pairs the system could have used entailment rules found in entailment and paraphrasing resources. As an example, the pair 565:

T - Soprano’s Square: Milan, Italy, home of the famed La Scala opera house, honored soprano Maria Callas on Wednesday when it renamed a new square after the diva.

H - La Scala opera house is located in Milan, Italy.

could be successfully solved using a paraphrase pattern such as $Y \text{ home of } X \iff X \text{ is located in } Y$, which can be found in (Lin and Pantel 2001). However, in order to use this kind of entailment rules, it would be necessary to extend the “single node” implementation of tree edit distance to address editing operations among subtrees.

Our participation in the RTE challenge served as a first test of our system. In the future, we plan to expand the system by searching for solutions for the mentioned problems and introducing *entailment* and *paraphrasing* resources.

References

- Dagan, I., Glickman, O. 2004 Generic applied modeling of language variability *In Proceedings of PASCAL Workshop on Learning Methods for Text Understanding and Mining* Grenoble
- Lin, D. 1998. Dependency-based evaluation of MINIPAR. *In Proceedings of the Workshop on Evaluation of Parsing Systems at LREC-98*. Granada, Spain.
- Lin, D. and Pantel, P. 2001. Discovery of inference rules for Question Answering. *Natural Language Engineering*, 7(4), pages 343-360.
- Monz, C. and de Rijke, M. 2001. Light-Weight Entailment Checking for Computational Semantics. *The third workshop on inference in computational semantics (ICoS-3)*.
- Punyakanok., V., Roth, D. and Yih, W., 2004 Mapping Dependencies Trees: An Application to Question Answering *Proceedings of AI & Math 2004*
- Ratnaparkhi, A. 1996 A Maximum Entropy Part-Of-Speech Tagger. *In proceeding of the Empirical Methods in Natural Language Processing Conference, May 17-18, 1996*
- Szpektor I., Tanev H., Dagan I., and Coppola B. 2004 Scaling Web-based Acquisition of Entailment Relations *In Proceedings of EMNLP-04 - Empirical Methods in Natural Language Processing, Barcelona, July 2004*
- K. Zhang K., Shasha D. 1990 Fast algorithm for the unit cost editing distance between trees. *Journal of algorithms*, vol. 11, p. 1245-1262, December 1990.