Similarity between Pairs of Co-indexed Trees for Textual Entailment Recognition

Fabio Massimo Zanzotto
DISCo
University Of Milan-Bicocca
Milano, Italy
zanzotto@disco.unimib.it

Alessandro Moschitti
DISP
University Of Rome "Tor Vergata"
Roma, Italy
moschitti@info.uniroma2.it

Abstract

In this paper we present a novel similarity between pairs of co-indexed trees to automatically learn textual entailment classifiers. We defined a kernel function based on this similarity along with a more classical intra-pair similarity. Experiments show an improvement of 4.4 absolute percent points over state-of-the-art methods.

1 Introduction

Recently, a remarkable interest has been devoted to textual entailment recognition (Dagan et al., 2005). The task requires to determine whether or not a text T entails a hypothesis H. As it is a binary classification task, it could seem simple to use machine learning algorithms to learn an entailment classifier from training examples. Unfortunately, this is not. The learner should capture the similarities between different pairs, (T', H') and (T'', H''), taking into account the relations between sentences within a pair. For example, having these two learning pairs:

$T_1 \Rightarrow H_1$	
T_1	*"At the end of the year, all solid companies pay dividends"*
H_1	*"At the end of the year, all solid insurance companies pay dividends."*

$T_1 \not\Rightarrow H_2$	
T_1	*"At the end of the year, all solid companies pay dividends"*
H_2	*"At the end of the year, all solid companies pay cash dividends."*

determining whether or not the following implication holds:

$T_3 \Rightarrow H_3$?	
T_3	*"All wild animals eat plants that have scientifically proven medicinal properties."*
H_3	*"All wild mountain animals eat plants that have scientifically proven medicinal properties."*

requires to detect that:

1. T_3 is structurally (and somehow lexically) similar to T_1 and H_3 is more similar to H_1 than to H_2;
2. relations between the sentences in the pairs (T_3, H_3) (e.g., T_3 and H_3 have the same noun governing the subject of the main sentence) are similar to the relations between sentences in the pairs (T_1, H_1) and (T_1, H_2).

Given this analysis we may derive that $T_3 \Rightarrow H_3$.

The example suggests that graph matching techniques are not sufficient as these may only detect the structural similarity between sentences of textual entailment pairs. An extension is needed to consider also if two pairs show compatible relations between their sentences.

In this paper, we propose to observe textual entailment pairs as pairs of syntactic trees with co-indexed nodes. This shuold help to cosider both the structural similarity between syntactic tree pairs and the similarity between relations among sentences within a pair. Then, we use this *cross-pair* similarity with more traditional *intra-pair* similarities (e.g., (Corley and Mihalcea, 2005)) to define a novel kernel function. We experimented with such kernel using Support Vector Machines on the Recognizing Textual Entailment (RTE) challenge test-beds. The comparative results show that (a) we have designed an effective way to automatically learn entailment rules

Workshop on TextGraphs, at HLT-NAACL 2006, pages 33–36,
New York City, June 2006.

from examples and (b) our approach is highly accurate and exceeds the accuracy of the current state-of-the-art models.

In the remainder of this paper, Sec. 2 introduces the cross-pair similarity and Sec. 3 shows the experimental results.

2 Learning Textual Entailment from examples

To carry out automatic learning from examples, we need to define a cross-pair similarity $K((T', H'), (T'', H''))$. This function should consider pairs similar when: (1) texts and hypotheses are structurally and lexically similar (*structural similarity*); (2) the relations between the sentences in the pair (T', H') are compatible with the relations in (T'', H'') (*intra-pair word movement compatibility*). We argue that such requirements could be met by augmenting syntactic trees with *placeholders* that co-index related words within pairs. We will then define a cross-pair similarity over these pairs of co-indexed trees.

2.1 Training examples as pairs of co-indexed trees

Sentence pairs selected as possible sentences in entailment are naturally co-indexed. Many words (or expressions) w_h in H have a referent w_t in T. These pairs (w_t, w_h) are called *anchors*. Possibly, it is more important that the two words in an anchor are related than the actual two words. The entailment could hold even if the two words are substitued with two other related words. To indicate this we co-index words associating *placeholders* with anchors. For example, in Fig. 1, 2" indicates the *(companies,companies)* anchor between T_1 and H_1. These placeholders are then used to augment tree nodes. To better take into account argument movements, placeholders are propagated in the syntactic trees following constituent heads (see Fig. 1).

In line with many other researches (e.g., (Corley and Mihalcea, 2005)), we determine these anchors using different similarity or relatedness dectors: the exact matching between tokens or lemmas, a similarity between tokens based on their edit distance, the derivationally related form relation and the verb entailment relation in WordNet, and, finally, a WordNet-based similarity (Jiang and Conrath, 1997). Each of these detectors gives a different weight to the anchor: the actual computed similarity for the last and 1 for all the others. These weights will be used in the final kernel.

2.2 Similarity between pairs of co-indexed trees

Pairs of syntactic trees where nodes are co-indexed with placeholders allow the design a cross-pair similarity that considers both the structural similarity and the intra-pair word movement compatibility.

Syntactic trees of texts and hypotheses permit to verify the structural similarity between pairs of sentences. Texts should have similar structures as well as hypotheses. In Fig. 1, the overlapping subtrees are in bold. For example, T_1 and T_3 share the subtree starting with **S** $\rightarrow$ **NP VP**. Although the lexicals in T_3 and H_3 are quite different from those T_1 and H_1, their bold subtrees are more similar to those of T_1 and H_1 than to T_1 and H_2, respectively. H_1 and H_3 share the production **NP** $\rightarrow$ **DT JJ NN NNS** while H_2 and H_3 do not. To decide on the entailment for (T_3,H_3), we can use the value of (T_1, H_1).

Anchors and placeholders are useful to verify if two pairs can be aligned as showing compatible intra-pair word movement. For example, (T_1, H_1) and (T_3, H_3) show compatible constituent movements given that the dashed lines connecting placeholders of the two pairs indicates structurally equivalent nodes both in the texts and the hypotheses. The dashed line between 3 and b links the main verbs both in the texts T_1 and T_3 and in the hypotheses H_1 and H_3. After substituting 3 to b and 2 to a, T_1 and T_3 share the subtree **S** $\rightarrow$ **NP** 2 **VP** 3. The same subtree is shared between H_1 and H_3. This implies that words in the pair (T_1, H_1) are correlated like words in (T_3, H_3). Any different mapping between the two anchor sets would not have this property.

Using the structural similarity, the placeholders, and the connection between placeholders, the overall similarity is then defined as follows. Let A' and A'' be the placeholders of (T', H') and (T'', H''), respectively. The similarity between two co-indexed syntactic tree pairs $K_s((T', H'), (T'', H''))$ is defined using a classical similarity between two trees $K_T(t_1, t_2)$ when the best alignment between the A' and A'' is given. Let C be the set of all bijective

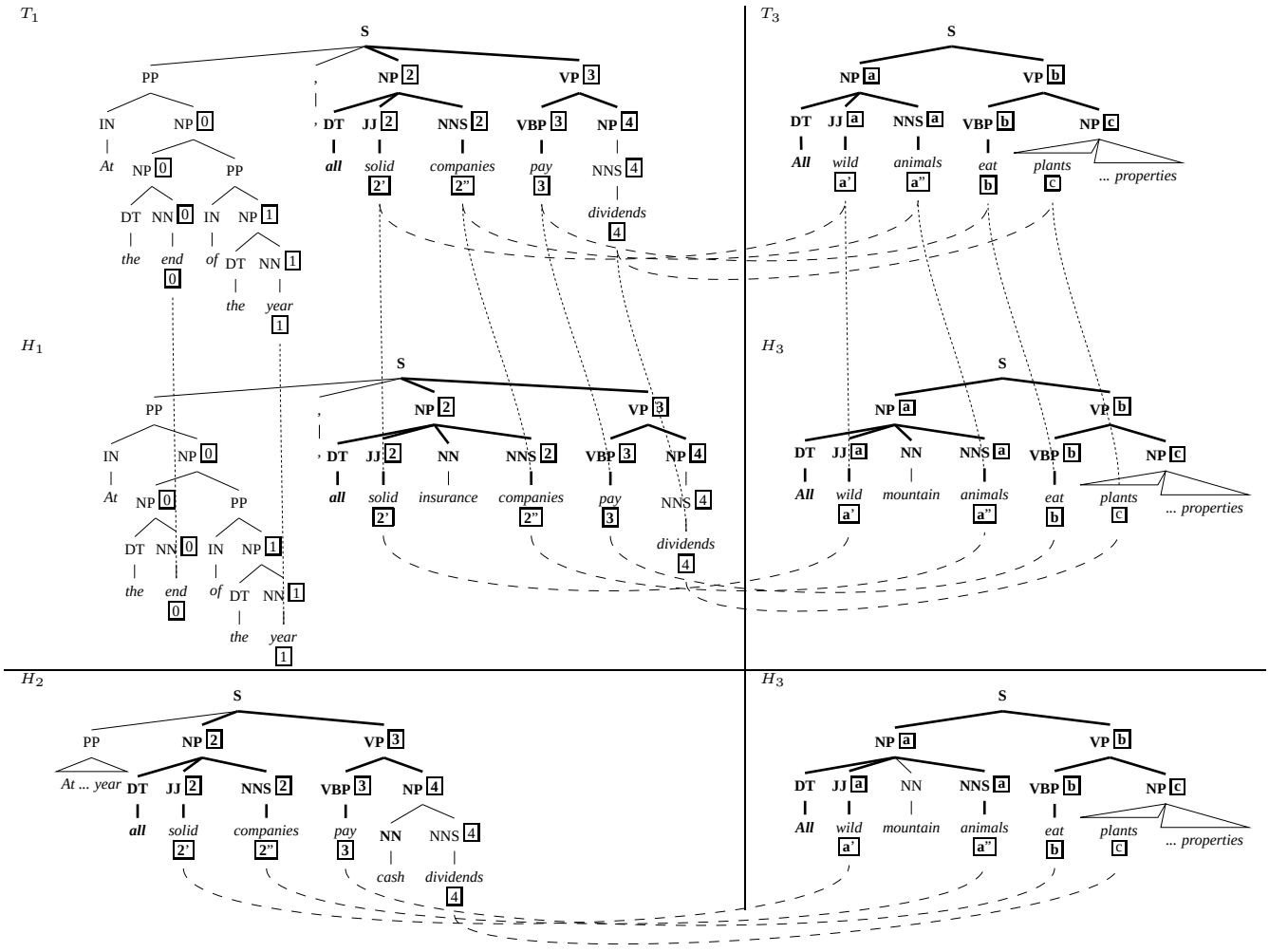

Figure 1: Relations between (T_1, H_1), (T_1, H_2), and (T_3, H_3).

mappings from $a' \subseteq A' : |a'| = |A''|$ to A'', an element $c \in C$ is a substitution function. The co-indexed tree pair similarity is then defined as:

$K_s((T', H'), (T'', H'')) =$
$max_{c \in C}(K_T(t(H', c), t(H'', i)) + K_T(t(T', c), t(T'', i))$

where (1) $t(S, c)$ returns the syntactic tree of the hypothesis (text) S with placeholders replaced by means of the substitution c, (2) i is the identity substitution and (3) $K_T(t_1, t_2)$ is a function that measures the similarity between the two trees t_1 and t_2.

2.3 Enhancing cross-pair syntactic similarity

As the computation cost of the similarity measure depends on the number of the possible sets of correspondences C and this depends on the size of the anchor sets, we reduce the number of *placeholders* used to represent the anchors. Placeholders will have the same name if these are in the same *chunk* both in the text and the hypothesis, e.g., the placeholders 2' and 2" are collapsed to 2.

3 Experimental investigation

The aim of the experiments is twofold: we show that (a) entailments can be learned from examples and (b) our kernel function over syntactic structures is effective to derive syntactic properties. The above goals can be achieved by comparing our cross-pair similarity kernel against (and in combination with) other methods.

3.1 Experimented kernels

We compared three different kernels: (1) the kernel $K_l((T', H'), (T'', H''))$ based on the intra-pair

Datasets	K_l	$K_l + K_t$	$K_l + K_s$
Train:$D1$ Test:$T1$	0.5888	0.6213	**0.6300**
Train:$T1$ Test:$D1$	0.5644	0.5732	**0.5838**
Train:$D2(50\%)'$ Test:$D2(50\%)''$	0.6083	0.6156	**0.6350**
Train:$D2(50\%)''$ Test:$D2(50\%)'$	0.6272	0.5861	**0.6607**
Train:$D2$ Test:$T2$	0.6038	0.6238	**0.6388**
Mean	0.5985 ($\pm$ 0.0235)	0.6040 ($\pm$ 0.0229)	**0.6297** ($\pm$ 0.0282)

Table 1: Experimental results

lexical similarity $sim_l(T, H)$ as defined in (Corley and Mihalcea, 2005). This kernel is defined as $K_l((T', H'), (T'', H'')) = sim_l(T', H') \times sim_l(T'', H'')$. (2) the kernel $K_l + K_s$ that combines our kernel with the lexical-similarity-based kernel; (3) the kernel $K_l + K_t$ that combines the lexical-similarity-based kernel with a basic tree kernel. This latter is defined as $K_t((T', H'), (T'', H'')) = K_T(T', T'') + K_T(H', H'')$. We implemented these kernels within SVM-light (Joachims, 1999).

3.2 Experimental settings

For the experiments, we used the Recognizing Textual Entailment (RTE) Challenge data sets, which we name as $D1$, $T1$ and $D2$, $T2$, are the development and the test sets of the first and second RTE challenges, respectively. $D1$ contains 567 examples whereas $T1$, $D2$ and $T2$ have all the same size, i.e. 800 instances. The positive examples are the 50% of the data. We produced also a random split of $D2$. The two folds are $D2(50\%)'$ and $D2(50\%)''$.

We also used the following resources: the Charniak parser (Charniak, 2000) to carry out the syntactic analysis; the `wn::similarity` package (Pedersen et al., 2004) to compute the Jiang&Conrath (J&C) distance (Jiang and Conrath, 1997) needed to implement the lexical similarity $sim_l(T, H)$ as defined in (Corley and Mihalcea, 2005); SVM-light-TK (Moschitti, 2004) to encode the basic tree kernel function, K_T, in SVM-light (Joachims, 1999).

3.3 Results and analysis

Table 1 reports the accuracy of different similarity kernels on the different training and test split described in the previous section. The table shows some important result.

First, as observed in (Corley and Mihalcea, 2005) the lexical-based distance kernel K_l shows an accuracy significantly higher than the random baseline, i.e. 50%. This accuracy (second line) is comparable with the best systems in the first RTE challenge (Dagan et al., 2005). The accuracy reported for the best systems, i.e. 58.6% (Glickman et al., 2005; Bayer et al., 2005), is not significantly far from the result obtained with K_l, i.e. 58.88%.

Second, our approach (last column) is significantly better than all the other methods as it provides the best result for each combination of training and test sets. On the "Train:$D1$-Test:$T1$" testbed, it exceeds the accuracy of the current state-of-the-art models (Glickman et al., 2005; Bayer et al., 2005) by about 4.4 absolute percent points (63% vs. 58.6%) and 4% over our best lexical similarity measure. By comparing the average on all datasets, our system improves on all the methods by at least 3 absolute percent points.

Finally, the accuracy produced by our kernel based on co-indexed trees $K_l + K_s$ is higher than the one obtained with the plain syntactic tree kernel $K_l + K_t$. Thus, the use of placeholders and co-indexing is fundamental to automatically learn entailments from examples.

References

Samuel Bayer, John Burger, Lisa Ferro, John Henderson, and Alexander Yeh. 2005. MITRE's submissions to the eu pascal rte challenge. In *Proceedings of the 1st Pascal Challenge Workshop*, Southampton, UK.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of the 1st NAACL*, pages 132–139, Seattle, Washington.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL RTE challenge. In *PASCAL Challenges Workshop*, Southampton, U.K.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. Web based probabilistic textual entailment. In *Proceedings of the 1st Pascal Challenge Workshop*, Southampton, UK.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the 10th ROCLING*, pages 132–139, Tapei, Taiwan.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods-Support Vector Learning*. MIT Press.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *proceedings of the ACL*, Barcelona, Spain.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proc. of 5th NAACL*, Boston, MA.