Experimenting a ”general purpose” textual entailment learner in AVE

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

Alessandro Moschitti
Department of Computer Science
University of Rome “Tor Vergata”
Rome, Italy
moschitti@info.uniroma2.it

Abstract

In this paper we present the use of a ”general purpose” textual entaiment recognizer in the Answer Validation Exercise (AVE) task. Our system has been developed to learn entailment rules from annotated examples. The main idea of the system is the cross-pair similirity measure we defined. This similarity allows us to define an implicit feature space using kernel functions in SVM learners. We experimented with our system using different training and testing sets: RTE data sets and AVE data sets. The comparative results show that entailment rules can be learned from data sets, e.g. RTE, that are different from AVE. Moreover, it seems that better results are obtained using more controlled training data (the RTE set) that less controlled ones (the AVE development set). Although, the high variability of the outcome prevents us to derive definitive conclusions, the results of our system show that our approach is quite promising and improvable in the future.

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: I.2.7 Natural Language Processing, I.2.6 Learning

General Terms

Measurement, Performance, Experimentation

Keywords

Question answering, Textual Entailment Recognition

1 Introduction

Textual entailment recognition is a common task performed in several natural language applications [8], e.g. Question Answering and Information Extraction. The Recognizing Textual Entaliment PASCAL Challenges [9, 2] fostered the development of several ”general purpose” textual entailment recognizers. CLEF 2006 instead provides an opportunity to show that those systems are useful for Question Answering. The voluntary exercise track aims to study the application of textual entailment recognition systems to the validation of correctness of answers given by QA systems. The basic idea is that once a pair answer/snippet is returned by a QA system, a hypothesis is built by turning the pair question/answer into an affirmative form. If the related text (a snippet or a document) semantically entails this hypothesis, then the answer is expected to be correct. The task of deciding this entailment is named here automatic Answer Validation Exercise (AVE).

We applied our entailment system [21], developed for the second automatic entailment recognition challenge (RTE) [2], to AVE. Our system has been shown to be one of the state-of-the-art systems on both RTE data sets [9, 2]. It determines whether or not a text T entails a hypothesis H by automatically learning rewriting rules from training positive and negative entailment pairs (T, H). For example given a text T_1: “*At the end of the year, all solid companies pay dividends.*” and two hypothesis:

a) H_1: “*At the end of the year, all solid <u>insurance</u> companies pay dividends*” and

b) H_2: "*At the end of the year, all solid companies pay cash dividends*",

we can built two examples: (T_1, H_1) which is an evidence of a true entailment (positive instance) and (T_1, H_2) which is a negative evidence.

Our system extract rules from them to solve apparently not related entailments. For example, given the following text and hypothesis:

$T_3 \Rightarrow H_3$?	
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."*

we note that T_3 is structurally (and somehow lexically similar) to T_1 and H_3 is more similar to H_1 than to H_2. Thus, from $T_1 \Rightarrow H_1$, we may extract rules to derive that $T_3 \Rightarrow H_3$.

The main idea of our model is that it relies not only on a *intra-pair* similarity between T and H but also on a *cross-pair* similarity between two pairs (T', H') and (T'', H''). The latter similarity measure along with a set of annotated examples allows the leaning model to automatically derive syntactic and lexical rules that can solve complex entailment cases.

In this paper, we experimented with our entailment recognition system [21] and the CLEF AVE. The comparative results show that entailment rules can be learned from data sets, e.g. RTE, that are different from AVE. Although, the high variability of the outcome prevents us to derive definitive conclusions, the results of our system show that our approach is quite promising and improvable in the future.

In the remainder of this paper, Sec. 2 illustrates the related work, Sec. 3 introduces the complexity of learning entailment rules from examples, Sec. 4 describes our models, Sec. 6 shows the experimental results, and, finally, Sec. 7 derives the conclusions.

2 Related work

Although the textual entailment recognition problem is not new, most of the automatic approaches have been proposed only recently. This has been mainly due to the RTE challenge events [9, 2]. In the following we report some of such researches.

A first class of methods defines measures of the distance or similarity between T and H either assuming the independence between words [7, 11] in a bag-of-word fashion or exploiting syntactic interpretations [16]. A pair (T, H) is then in entailment when $sim(T, H) > \alpha$. These approaches can hardly determine whether the entailment holds in the examples of the previous section. From the point of view of bag-of-word methods, the pairs (T_1, H_1) and (T_1, H_2) have both the same intra-pair similarity since the sentences of T_1 and H_1 as well as those of T_1 and H_2 differ by a noun, *insurance* and *cash*, respectively. At syntactic level, also, we cannot capture the required information as such nouns are both noun modifiers: *insurance* modifies *companies* and *cash* modifies *dividends*.

A second class of methods can give a solution to the previous problem. These methods generally combine a similarity measure with a set of possible transformations $\mathcal{T}$ applied over syntactic and semantic interpretations. The entailment between T and H is detected when there is a transformation $r \in \mathcal{T}$ so that $sim(r(T), H) > \alpha$. These transformations are logical rules in [3] or sequences of allowed *rewrite rules* in [10]. The disadvantage is that such rules have to be manually designed. Moreover, they generally model better positive implications than negative ones and they do not consider errors in syntactic parsing and semantic analysis.

3 Challenges in learning from examples

In the introductory section, we have shown that, to carry out automatic learning from examples, we need to define a cross-pair similarity measure. Its definition is not straightforward as it should detect whether two pairs (T', H') and (T'', H'') realize the same *rewrite rules*. This measure should consider pairs similar when: (1) T' and H' are structurally similar to T'' and H'', respectively and (2) the lexical relations within the pair (T', H') are compatible with those in (T'', H''). Typically, T and H show a certain degree of overlapping, thus, lexical relations (e.g., between the same words) determine *word movements* from T to H (or vice versa). This is important to model the syntactic/lexical similarity between example pairs.

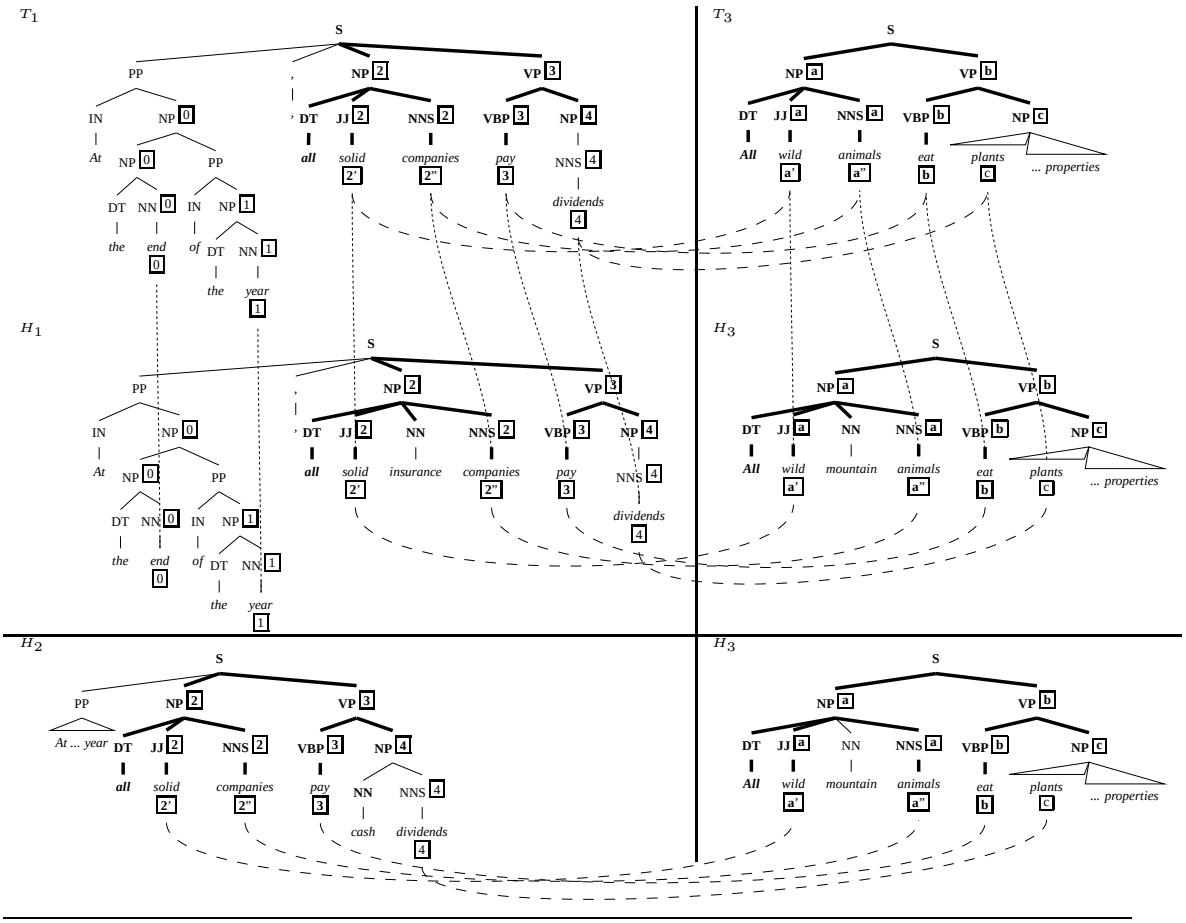

Figure 1: Relations between (T_1, H_1), (T_1, H_2), and (T_3, H_3).

Indeed, if we encode such movements in the syntactic parse trees of texts and hypotheses, we can use interesting similarity measures defined for syntactic parsing, e.g., the tree kernel devised in [6].

To consider structural and lexical relation similarity, we augment syntactic trees with *placeholders* which identify linked words. More in detail:

- We detect links between words w_t in T that are equal, similar, or semantically dependent on words w_h in H. We call *anchors* the pairs (w_t, w_h) and we associate them with *placeholders*. For example, in Fig. 1, the placeholder 2" indicates the *(companies,companies)* anchor between T_1 and H_1. This allows us to derive the word movements between text and hypothesis.
- We align the trees of the two texts T' and T'' as well as the tree of the two hypotheses H' and H'' by considering the *word movements*. We find a correct mapping between placeholders of the two hypothesis H' and H'' and apply it to the tree of H'' to substitute its placeholders. The same mapping is used to substitute the placeholders in T''. This mapping should maximize the structural *similarity* between the four trees by considering that placeholders augment the node labels. Hence, the cross-pair similarity computation is reduced to the tree similarity computation.

The above steps define an effective cross-pair similarity that can be applied to the example in Fig. 1: T_1 and T_3 share the subtree in bold starting with **S** $\rightarrow$ **NP VP**. The lexicals in T_3 and H_3 are quite different from those T_1 and H_1, but we can rely on the structural properties expressed by their bold subtrees. These are more similar to the subtrees of T_1 and H_1 than those of T_1 and H_2, respectively. Indeed, H_1 and H_3 share the production **NP** $\rightarrow$ **DT JJ NN NNS** while H_2 and H_3 do not. Consequently, to decide if (T_3,H_3) is a valid entailment, we should rely on the decision made for (T_1, H_1). Note also that the dashed lines connecting placeholders of two texts (hypotheses) indicate structurally equivalent nodes. For instance, 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 with b and 2 with a, we can detect if T_1 and T_3 share the bold subtree **S** $\rightarrow$ **NP** 2 **VP** 3. As such subtree is shared also by H_1 and H_3, the words within the pair (T_1, H_1) are correlated similarly to the words in (T_3, H_3).

The above example emphasizes that we need to derive the *best* mapping between placeholder sets. It

can be obtained as follows: let A' and A'' be the placeholders of (T', H') and (T'', H''), respectively, without loss of generality, we consider $|A'| \geq |A''|$ and we align a subset of A' to A''. The best alignment is the one that maximizes the syntactic and lexical overlapping of the two subtrees induced by the aligned set of anchors.

More precisely, let C be the set of all bijective mappings from $a' \subseteq A' : |a'| = |A''|$ to A'', an element $c \in C$ is a substitution function. We define as the best alignment the one determined by

$$c_{max} = argmax_{c \in C}(K_T(t(H', c), t(H'', i)) + K_T(t(T', c), t(T'', i)) \quad (1)$$

where (a) $t(S, c)$ returns the syntactic tree of the hypothesis (text) S with placeholders replaced by means of the substitution c, (b) i is the identity substitution and (c) $K_T(t_1, t_2)$ is a function that measures the similarity between the two trees t_1 and t_2 (for more details see Sec. 4.2). For example, the c_{max} between (T_1, H_1) and (T_3, H_3) is $\{(2', a'), (2'', a''), (3, b), (4, c)\}$.

4 Similarity Models

In this section we describe how anchors are found at the level of a single pair (T, H) (Sec. 4.1). The anchoring process gives the direct possibility of implementing an inter-pair similarity that can be used as a baseline approach or in combination with the cross-pair similarity. This latter will be implemented with tree kernel functions over syntactic structures (Sec. 4.2).

4.1 Anchoring and Lexical Similarity

The algorithm that we design to find the anchors is based on similarity functions between words or more complex expressions. Our approach is in line with many other researches (e.g., [7, 11]).

Given the set of content words (verbs, nouns, adjectives, and adverbs) W_T and W_H of the two sentences T and H, respectively, the set of anchors $A \subset W_T \times W_H$ is built using a similarity measure between two words $sim_w(w_t, w_h)$. Each element $w_h \in W_H$ will be part of a pair $(w_t, w_h) \in A$ if:

1. $sim_w(w_t, w_h) \neq 0$
2. $sim_w(w_t, w_h) = \max_{w'_t \in W_T} sim_w(w'_t, w_h)$

According to these properties, elements in W_H can participate in more than one anchor and conversely more than one element in W_H can be linked to a single element $w \in W_T$.

The similarity $sim_w(w_t, w_h)$ can be defined using different indicators and resources. First of all, two words are maximally similar if these have the same surface form $w_t = w_h$. Second, we can use one of the WordNet [17] similarities indicated with $d(l_w, l_{w'})$ (in line with what was done in [7]) and different relation between words such as the lexical entailment between verbs (Ent) and derivationally relation between words (Der). Finally, we use the edit distance measure $lev(w_t, w_h)$ to capture the similarity between words that are missed by the previous analysis for misspelling errors or for the lack of derivationally forms not coded in WordNet.

As result, given the syntactic category $c_w \in \{noun, verb, adjective, adverb\}$ and the lemmatized form l_w of a word w, the similarity measure between two words w and w' is defined as follows:

$$sim_w(w, w') = \begin{cases} 1 & \textbf{if } w = w' \vee \\ & l_w = l_{w'} \wedge c_w = c_{w'} \vee \\ & ((l_w, c_w), (l_{w'}, c_{w'})) \in Ent \vee \\ & ((l_w, c_w), (l_{w'}, c_{w'})) \in Der \vee \\ & lev(w, w') = 1 \\ d(l_w, l_{w'}) & \textbf{if } c_w = c_{w'} \wedge d(l_w, l_{w'}) > 0.2 \\ 0 & \textbf{otherwise} \end{cases} \quad (2)$$

It is worth noticing that, the above measure is not a *pure* similarity measure as it includes the entailment relation that does not represent synonymy or similarity between verbs. To emphasize the contribution of each used resource, in the experimental section, we will compare Eq. 2 with some versions that exclude some word relations.

The above word similarity measure can be used to compute the similarity between T and H. In line with [7], we define it as:

$$s(T,H) = \frac{\sum_{(w_t,w_h)\in A} sim_w(w_t,w_h) \times idf(w_h)}{\sum_{w_h \in W_H} idf(w_h)} \quad (3)$$

where $idf(w)$ is the inverse document frequency of the word w.

From the above intra-pair similarity, we can obtain the baseline *cross-pair* similarity based on only lexical information:

$$K_{lex}((T',H'),(T'',H'')) = s(T',H') \times s(T'',H'') \quad (4)$$

In the next section we define a novel cross-pair similarity that takes into account syntactic evidence by means of tree kernel functions.

4.2 Cross-pair syntactic kernels

Section 3 has shown that to measure the syntactic similarity between two pairs, (T',H') and (T'',H''), we should capture the number of common subtrees between texts and hypotheses that share the same anchoring scheme. The best alignment between anchor sets, i.e. the best substitution c_{max}, can be found with Eq. 1. As the corresponding maximum quantifies the *alignment degree*, we could define a cross-pair similarity as follows:

$$K_{struct}((T',H'),(T'',H'')) = \max_{c\in C} \big(K_T(t(H',c),t(H'',i)) + K_T(t(T',c),t(T'',i)\big), \quad (5)$$

where as $K_T(t_1,t_2)$ we use the tree kernel function defined in [6]. This evaluates the number of subtrees shared by t_1 and t_2, thus defining an implicit substructure space.

Formally, given a subtree space $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$, the indicator function $I_i(n)$ is equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over t_1 and t_2 is $K_T(t_1,t_2) = \sum_{n_1\in N_{t_1}} \sum_{n_2\in N_{t_2}} \Delta(n_1,n_2)$, where N_{t_1} and N_{t_2} are the sets of the t_1's and t_2's nodes, respectively. In turn $\Delta(n_1,n_2) = \sum_{i=1}^{|\mathcal{F}|} \lambda^{l(f_i)} I_i(n_1) I_i(n_2)$, where $0 \le \lambda \le 1$ and $l(f_i)$ is the number of levels of the subtree f_i. Thus $\lambda^{l(f_i)}$ assigns a lower weight to larger fragments. When $\lambda = 1$, Δ is equal to the number of common fragments rooted at nodes n_1 and n_2. As described in [6], Δ can be computed in $O(|N_{t_1}| \times |N_{t_2}|)$.

The K_T function has been proven to be a valid kernel, i.e. its associated *Gram* matrix is positive-semidefinite. Some basic operations on kernel functions, e.g. the sum, are closed with respect to the set of valid kernels. Thus, if the maximum held such property, Eq. 5 would be a valid kernel and we could use it in kernel based machines like SVMs. Unfortunately, a counterexample illustrated in [4] shows that the *max* function does not produce valid kernels in general.

However, we observe that: (1) $K_{struct}((T',H'),(T'',H''))$ is a symmetric function since the set of transformation C are always computed with respect to the pair that has the largest anchor set; (2) in [12], it is shown that when kernel functions are not positive semidefinite, SVMs still solve a data separation problem in pseudo Euclidean spaces. The drawback is that the solution may be only a local optimum. Therefore, we can experiment Eq. 5 with SVMs and observe if the empirical results are satisfactory. Section 6 shows that the solutions found by Eq. 5 produce accuracy higher than those evaluated on previous automatic textual entailment recognition approaches.

5 Refining cross-pair syntactic similarity

In the previous section we have defined the intra and the cross pair similarity. The former does not show relevant implementation issues whereas the latter should be optimized to favor its applicability with SVMs. The Eq. 5 improvement depends on two factors: (1) its computation complexity; (2) the pruning of irrelevant information in large syntactic trees.

5.1 Controlling the computational cost

The computational cost of cross-pair similarity between two tree pairs (Eq. 5) depends on the size of C. This is combinatorial in the size of A' and A'', i.e. $|C| = (|A'| - |A''|)!|A''|!$ if $|A'| \geq |A''|$. Thus we should keep the sizes of A' and A'' reasonably small.

To reduce the number of placeholders, we consider the notion of *chunk* defined in [1], i.e., *not recursive kernels* of noun, verb, adjective, and adverb phrases. When placeholders are in a single chunk both in the text and hypothesis we assign them the same name. For example, Fig. 1 shows the placeholders 2' and 2" that are substituted by the placeholder 2. The placeholder reduction procedure also gives the possibility of resolving the ambiguity still present in the anchor set A (see Sec. 4.1). A way to eliminate the ambiguous anchors is to select the ones that reduce the final number of placeholders.

5.2 Pruning irrelevant information in large *text* trees

Often only a portion of the parse trees is relevant to detect entailments. For instance, let us consider the following pair from the RTE 2005 corpus:

$T \Rightarrow H$	(id: 929)
T	"***Ron Gainsford, chief executive of the TSI,*** *said: "It is a major concern to us that parents could be unwittingly exposing their children to the risk of sun damage, thinking they are better protected than they actually are."*
H	"*Ron Gainsford is the chief executive of the TSI.*"

Only the bold part of T supports the implication; the rest is useless and also misleading: if we used it to compute the similarity it would reduce the importance of the relevant part. Moreover, as we normalize the syntactic tree kernel (K_T) with respect to the size of the two trees, we need to focus only on the part relevant to the implication.

The anchored leaves are good indicators of relevant parts but also some other parts may be very relevant. For example, the function word *not* plays an important role. Another example is given by the word *insurance* in H_1 and *mountain* in H_3 (see Fig. 1). They support the implication $T_1 \Rightarrow H_1$ and $T_1 \Rightarrow H_3$ as well as *cash* supports $T_1 \not\Rightarrow H_2$. By removing these words and the related structures, we cannot determine the correct implications of the first two and the incorrect implication of the second one. Thus, we keep all the words that are immediately related to relevant constituents.

The reduction procedure can be formally expressed as follows: given a syntactic tree t, the set of its nodes $N(t)$, and a set of anchors, we build a tree t' with all the nodes N' that are anchors or ancestors of any anchor. Moreover, we add to t' the leaf nodes of the original tree t that are direct children of the nodes in N'. We apply such procedure only to the syntactic trees of texts before the computation of the kernel function.

6 Experimental investigation

The experiments aim at determining if our system can learn rules required to solve the entailment cases contained in the AVE data set. Although, we have already shown that our system can learn entailment [9, 2], the task here appears to be more complex as: (a) texts are automatically built from answers and questions; this necessarily introduces some degree of noise; and (b) often question answering systems provide a correct answer but the supporting text is not adequate to carry out a correctness inference, e.g. a lot background knowledge is required or the answer was selected by chance.

Our approach to study the above points is to train and experiment with our system and several data sets proposed in AVE as well as RTE1 and RTE2. The combination of training and testing based on such sets can give an indication on the learnability of general rules valid for different domain and different applications.

6.1 Experimental settings

For the experiments, we used the following data sets:

Training set	Test sets	Trade-off parameter		
		j=1	j=10	j=0.9
AVE_a	AVE_b	11.55	35.36	-
AVE_b	AVE_a	x	31.85	-
$AVE_b \cup RTE1$	AVE_a	12.20	37.14	-
$AVE_b \cup RTE1 \cup RTE2$	AVE_a	28.57	35.89	-
$AVE_b \cup RTE2$	AVE_a	25.68	38.98	-
$AVE_a \cup RTE1$	AVE_b	22.57	32.05	-
$AVE_a \cup RTE1 \cup RTE2$	AVE_b	31.76	30.85	-
$AVE_a \cup RTE2$	AVE_b	34.07	32.38	-
$RTE1$	AVE_a	39.81	30.64	-
$RTE1$	AVE_b	35.58	28.31	-
$RTE1 \cup RTE2$	AVE_a	38.27	31.58	40.85
$RTE1 \cup RTE2$	AVE_b	33.42	28.29	36.20
$RTE2$	AVE_a	37.46	33.04	-
$RTE2$	AVE_b	35.57	29.72	-

Table 1: F1 measure of our entailment system trained with data from RTE1, RTE2 and AVE tested on the AVE split (AVE_a and AVE_b).

- $RTE1$ and $RTE2$, i.e. the sets (development and test data) of the first [9] and second [2] challenges, respectively. $RTE1$ contains 1,367 examples whereas $RTE2$ contain 1,600 instances. The positive and negative examples are equally distributed in the collection, i.e. 50% of the data.
- AVE_a and AVE_b come from a random split of the AVE development set, we created it to homogeneously learn and test our model on the AVE data. The AVE development set contains 2870 instances. Here, the positive and negative examples are not equally distributed. It contains 436 positive 2434 negative examples.

We also created new sets by merging groups of the above four collections. For example, $AVE_a \cup RTE1 \cup RTE2$ stands for the set obtained as union of AVE_a, $RTE1$ and $RTE2$. Moreover, to implement our model (described in sections 4 and 5), we used the following resources:

- The Charniak parser [5] and the `morpha` lemmatiser [18] to carry out the syntactic and morphological analysis.
- WordNet 2.0 [17] to extract both the verbs in entailment, Ent set, and the derivationally related words, Der set.
- The `wn::similarity` package [20] to compute the Jiang&Conrath (J&C) distance [14] as in [7]. This is one of the best figure method which provides a similarity score in the $[0, 1]$ interval. We used it to implement the $d(l_w, l_{w'})$ function.
- A selected portion of the British National Corpus[1] to compute the inverse document frequency (idf). We assigned the maximum idf to words not found in the BNC.
- SVM-light-TK[2] [19] which encodes the basic tree kernel function, K_T, in SVM-light [15]. We used such software to implement the overall kernel $K_{overall} = K_{lex} + K_{struct}$ (see equations 4 and 5). In all the experiments we used $K_{overall}$ which combines the lexical and structural cross similarities.

6.2 Results and analysis

Table 1 reports the results of our system trained with data from RTE1, RTE2 and AVE tested on the AVE split (AVE_a and AVE_b). Columns 1 and 2 denote the data set used for training and testing, respectively whereas column 2, 3 and 4 illustrated the F1 measure of the systems with respect to 3 different values of the j parameters, 1, 10 and 0.9, respectively. Such parameter tunes the trade-off between Precision and Recall. Higher values cause the system to retrieve more positive examples. When the system has a Recall of 0, the table shows the "x" symbol while the symbol "-" indicates that the experiment has not been performed.

Following aspects should be noted:

[1] http://www.natcorp.ox.ac.uk/

[2] SVM-light-TK is available at `http://ai-nlp.info.uniroma2.it/moschitti/`

- Training on AVE_a and testing on AVE_b provides almost 4% more in F1 than training on AVE_b and testing on AVE_a. This suggests the high variability of the results due to few training data; also testified by the high impact of the j parameter (about 24% of difference between $j = 1$ and $j = 10$).
- If we add the examples from the RTE challenges to the AVE_b training data, we obtain a good improvement, e.g. the system trained on $AVE_b \cup RTE2$ improves the one trained on AVE_b of about 7% (38.98% vs. 31.85%). Adding $RTE1$ to the training data causes a decrease. This could be explained by the high impact of parameters. It is possible that the *good* setting for $AVE_b \cup RTE2$ is not very good for $AVE_b \cup RTE1 \cup RTE2$.
- Training on AVE_a and RTE data sets seems not helpful as the result using only AVE_a is higher, e.g. 35.36% vs. 32.38%.
- Finally, training on $RTE1$ provides higher performance than training on $RTE2$ on both AVE_a and AVE_b test sets (see rows 10 and 11 vs 14 and 15). Moreover, their combined use ($RTE1 \cup RTE2$) is helpful only if we select an opportune parameter j=0.9. This leads to the highest performance on AVE_a and AVE_b, i.e. 40.85% and 36.20%, respectively.

Given these preliminary results, we decided to use the best model obtained on $RTE1 \cup RTE2$ to generate data of our CLEF submission. Moreover, as the AVE test set may have been statistically similar to the development set, we also submitted a run of the model trained on $AVE_a \cup AVE_b$. The official results were 39.95% and 36.69%, respectively. These are quite in line with the analogous experiments shown in Table 1, i.e. training on $RTE1 \cup RTE2$ and testing on AVE_a (40.85%) and training on AVE_a and testing on AVE_b (35.36).

6.2.1 Qualitative analysis

The system we presented strongly uses syntactic interpretations of the example pairs. Then, its major bottleneck is the standard AVE process used to produce the affirmative form of the question given the answer provided by a the QA system. This process frequently generates ungrammatical sentences. The problem is clear just reading the first instances of the AVE development set. We report hereafter some of these examples. Each table reports the original question (Q), the text snippet (T), and the affirmative form of the question used as hypothesis (H).

$T \Rightarrow H$ (id: 1)	
Q	"*When did Nixon resign?*"
T	"*August, 1974 – Nixon resigns.*"
H	"*Nixon resigned in 1974 – Nixon*"

$T \Rightarrow H$ (id: 2)	
Q	"*What year was Halley's comet visible?*"
T	"*[...] 1909 Halley's comet sighted from Cambridge Observatory. 1929 [...]*"
H	"*In 1909 Halley was Halley's comet visible "*

$T \Rightarrow H$ (id: 6)	
Q	"*Who is Juan Antonio Samaranch?*"
T	"*International Olympic Committee President Juan Antonio Samaranch came strongly to the defense of China's athletes, [...]*"
H	"*Juan Antonio Samaranch is International Olympic Committee President Juan Antonio Samaranch came strongly to the defense of China 's athletes "*

We can observe that these examples have highly ungrammatical hypothesis. In the example (id 1), *Nixon* is repeated at the end of H. In the example (id 2), *Halley* is used as subject and as predicate. Finally, in the example (id 6) there is a large part of the hypothesis that is unnecessary and creates an ungrammatical sentence.

7 Conclusions

In this paper, we experimented with our entailment system [21] and the CLEF AVE. The comparative results show that entailment rules can be learned from data sets, e.g. RTE, that are different from AVE.

The experiments show that few training examples and data sparseness produce a high variability of the results. In this scenario the parameterization is very critical and necessitates of accurate cross-validation techniques. The AVE results also show that our model can learn entailments from the RTE data sets (with a higher F1 than using only AVE data). This suggests that there are some general rules, valid cross domains and collections. The importance of such rules is still more evident if we consider that the distribution of positive and negative examples in the RTE and AVE data sets is quite different. This usually prevents statistical learning algorithms to carry out a correct generalization of the data.

In the future, we would like to carry out a throughout parameterization and continue investigating approaches to exploit data from difference sources of entailments.

References

[1] Steven Abney. Part-of-speech tagging and partial parsing. In G.Bloothooft K.Church, S.Young, editor, *Corpus-based methods in language and speech*. Kluwer academic publishers, Dordrecht, 1996.

[2] Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The II PASCAL RTE challenge. In *PASCAL Challenges Workshop*, Venice, Italy, 2006.

[3] Johan Bos and Katja Markert. Recognising textual entailment with logical inference. In *Proc. of HLT-EMNLP Conference*, pages 628–635, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[4] S. Boughorbel, J-P. Tarel, and F. Fleuret. Non-mercer kernel for svm object recognition. In *Proceedings of BMVC 2004*, pages 137–146, 2004.

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

[6] Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL02*, 2002.

[7] Courtney Corley and Rada Mihalcea. 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 2005. Association for Computational Linguistics.

[8] Ido Dagan and Oren Glickman. Probabilistic textual entailment: Generic applied modeling of language variability. In *Proceedings of the Workshop on Learning Methods for Text Understanding and Mining*, Grenoble, France, 2004.

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

[10] Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. An inference model for semantic entailment in natural language. In *Proc. of The PASCAL RTE Challenge Workshop*, Southampton, U.K, 2005.

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

[12] Bernard Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Trans Pattern Anal Mach Intell*, 27(4):482–92, Apr 2005.

[13] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 15th CoLing*, Nantes, France, 1992.

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

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

[16] Milen Kouylekov and Bernardo Magnini. Tree edit distance for textual entailment. In *Proc. of the RANLP-2005*, Borovets, Bulgaria, 2005.

[17] George A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

[18] Guido Minnen, John Carroll, and Darren Pearce. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223, 2001.

[19] Alessandro Moschitti. Making tree kernels practical for natural language learning. In *Proceedings of EACL'06*, Trento, Italy, 2006.

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

[21] Fabio Massimo Zanzotto and Alessandro Moschitti. Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 401–408, Sydney, Australia, July 2006. Association for Computational Linguistics.