Experimenting a "general purpose" textual entailment learner in AVE

Fabio Massimo Zanzotto and Alessandro Moschitti

ART Group, DISP, University of Rome "Tor Vergata", Rome, Italy
{zanzotto,moschitti}@info.uniroma2.it,
WWW home page: http://ai-nlp.info.uniroma2.it

Abstract. In this paper we present the use of a "general purpose" textual entailment recognizer in the Answer Validation Exercise (AVE) task. Our system is designed to learn entailment rules from annotated examples. Its main feature is the use of Support Vector Machines (SVMs) with kernel functions based on cross-pair similarity between entailment pairs. We experimented with our system using different training sets: RTE and AVE data sets. The comparative results show that entailment rules can be learned. Although, the high variability of the outcome prevents us to derive definitive conclusions, the results show that our approach is quite promising and improvable in the future.

1 Introduction

Textual entailment recognition is a common task performed in several natural language processing applications [1], e.g. Question Answering and Information Extraction. The Recognizing Textual Entailment (RTE) PASCAL Challenges [2, 3] fostered the development of several "general purpose" textual entailment recognizers. The Answer Validation Exercise (AVE) instead provides an opportunity to show that such recognizers are useful for Question Answering.

We applied our textual entailment recognition system [4], developed for the second RTE challenge [3], to AVE. Our system has been shown to achieve state-of-the-art results on both RTE data sets [2, 3]. It determines whether or not a text T entails a hypothesis H by automatically learning rewriting rules from positive and negative training instances of entailment pairs (T, H). For example given a text T_1: "*At the end of the year, all solid companies pay dividends.*" and two hypotheses:

a) H_1: "*At the end of the year, all solid insurance 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 a true entailment (positive instance) and (T_1, H_2) which is a negative one. 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 textual entailment recognition system [4] and the CLEF AVE. We show that entailment recognition rules can be learnt from annotated examples. In the remainder of this paper, Sec. 2 describes our model in a glance, Sec. 3 shows the experimental results, and, finally, Sec. 4 derives the conclusions.

2 Our textual entailment learner in a glance

In this section we will shortly introduce the main idea of our method, which is fully described in [4].

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. The pairs (w_t, w_h) are called *anchors*, where different relations between w_t and w_h may be hold. Therefore, entailment could hold even if the two words are substituted with two other related words. To indicate the relatedness property, we co-index words with *placeholders* which in turn are associated 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

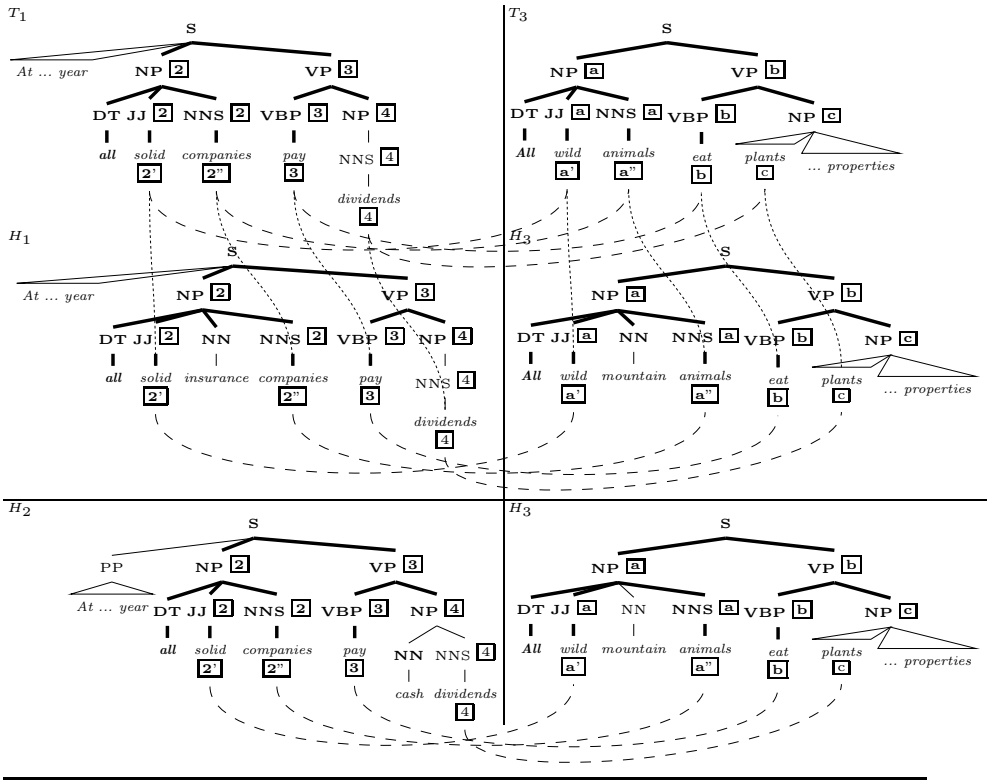

Fig. 1. Relations between (T_1, H_1), (T_1, H_2), and (T_3, H_3).

account argument movements, placeholders are propagated in the syntactic trees following constituent heads (see Fig. 1).

In line with many other researches (e.g., [5]), we determine these anchors using different similarity or relatedness models: 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 [6]. Each of these detectors gives a different weight to the anchor: 0/1 for the first and the similarity value 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 us to design a cross-pair similarity based on both the structural similarity and the intra-pair word movement compatibility.

Syntactic trees of texts and hypotheses are useful to detect 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 decision made on (T_1, H_1).

Anchors and placeholders are useful to verify if two pairs can be aligned by checking the compatibility between intra-pair word movements. 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 indicate 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 $K_s((T', H'), (T'', H''))$ is then defined on a kernel $K_T(st_1, st_2)$ (as the one described in [7]) that measures the similarity between two syntactic trees st_1 and st_2. For more details on the kernel function see [4].

3 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 rules [2, 3], 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 in the RTE challenges. The different training based on such sets can give an indication on the learnability of general rules valid for different domain and different applications.

3.1 Experimented kernels

We compared three different kernels: (1) the kernel $K_l((T', H'), (T'', H'')) = sim_l(T', H') \times sim_l(T'', H'')$ which is based on the intra-pair lexical similarity $sim_l(T, H)$ described in [5]. (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. The latter is defined as $K_t((T', H'), (T'', H'')) = K_T(T', T'') + K_T(H', H'')$.

3.2 Experimental settings

For the experiments, we used the following data sets:

- RTE, i.e. the set derived using all the data (development and test data) of the first [2] and second [3] RTE challenges. Respectively, the dataset of the first RTE contains 1,367 examples whereas the one of the second contains 1,600 instances. The positive and negative examples are equally distributed in the collection, i.e. 50% of the data.
- AVE, i.e., the AVE development set. The AVE development set contains 2870 instances. Here, the positive and negative examples are not equally distributed. It contains 436 positive and 2434 negative examples.
- AVE_{test}, i.e., the AVE testing set. The AVE test set contains 2088 instances. It contains 198 positive 1048 and negative examples. The remaining 842 examples are unknown (see the AVE task overall description [8] for more details) and are not used for evaluation purposes.

We also created a new set $AVE \cup RTE$ by merging groups of the two of the above collections.

In the remainder of this section, we describe the resources used to implement our system.

- The Charniak parser [9] and the `morpha` lemmatiser [10] to carry out the syntactic and morphological analysis.
- WordNet 2.0 [11] to extract both the verbs in entailment, *Ent* set, and the derivationally related words, *Der* set.
- The `wn::similarity` package [12] to compute the Jiang&Conrath (J&C) distance [6] as in [5]. 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] [13] which encodes the basic tree kernel function, K_T, in SVM-light [14]. We used such software to implement the kernels K_l, K_s, and K_t and their linear combinations.

3.3 Results and analysis

Table 1 reports the results of our system trained with data from RTE and AVE development and tested on the AVE test set (AVE_{test}). Column 1 denotes the data set used for training. Column 2 illustrates the value of the j parameter, respectively. Such parameter tunes the trade-off between Precision and Recall. Higher values cause the system to retrieve more positive examples. We used

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

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

Model		K_l			$K_s + K_l$			$K_t + K_l$		
Train	j	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
AVE	10	0.2512	0.8232	0.3849	**0.3858**	**0.4949**	**0.4336**	0.3979	0.3838	0.3907
RTE	0.9	0.2916	0.8232	0.4307	**0.2944**	**0.7121**	**0.4166**	0.2872	0.7121	0.4093
	1	0.2820	0.8232	0.4201	0.3008	0.7778	0.4338	0.2729	0.7525	0.4005
	10	0.1589	1.0000	0.2742	0.2131	0.9343	0.3470	0.2305	0.9697	0.3725
$RTE \cup AVE$	0.9	0	0.0000	0	0.3431	0.4747	0.3983	0.2971	0.5101	0.3755
	1	0	0.0000	0	0.3399	0.6061	0.4355	0.2844	0.6364	0.3931
	10	0.1589	1.0000	0.2742	0.2290	0.8384	0.3597	0.2408	0.8939	0.3794

Table 1. The results of the different models over the test set AVE_{test}.

three values: 0.9, 1, and 10. Row 1 indicates the kernel that has been used for the experiments. The Precision, the Recall, and the F-measure are computed on the positive pairs (yes pairs) as discussed in the AVE task description [8]. When the system has a Recall of 0, we assign Precision and the F-Measure to 0. Results for the systems trained on the AVE with j equals to 0.9 and 1 are not in the table as they all show a 0 Recall. The revised[3] results of the two runs submitted are in bold. ZNZ-TV_1 corresponds to the $K_l + K_s$ system trained on the AVE set with $j = 10$ whereas ZNZ-TV_2 corresponds to the $K_l + K_s$ system trained on the RTE set with $j = 0.9$.

The following aspects should be noted:

- When learning on the AVE set, the small amount of positive examples limits the possibility of learning the positive entailments. The resulting model classifies always as negative example. The only way of extracting positive rules is to give more importance to positive examples changing the j trade-off parameter. Only with the value 10, a model retrieving positive examples is learnt.
- The model $K_l + K_s$ is generally better than the model $K_l + K_t$. For example, for the submitted system trained on AVE, the difference is of about 5 points in term of F-Measure. This indicates that placeholders are very relevant for learning entailment rules.
- The simple model K_l learnt on the RTE has an F-Measure comparable with those obtained by the best model, i.e. 0.4336 of $K_l + K_s$ system trained on the AVE set. This odd result is generated by the high variability of system accuracy, e.g. slightly different parameterization produce very different results.

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

[3] Results presented here differ from the official results. At the time of the submission, the K_s part of the combined kernel $K_l + K_s$ didn't work properly for a formatting error of the syntactic trees. Basically, the systems submitted were the systems based on K_l with some noise.

the original question (Q), the text snippet (T), and the affirmative form of the question used as hypothesis (H).

$T \Rightarrow H$ (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 hypotheses. 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.

4 Conclusions

In this paper, we experimented with our entailment system [4] and the CLEF AVE. The comparative results show that entailment rules can be learned from the data set 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. In the future, we would like to carry out a throughout parameterization and continue investigating approaches to exploit data from difference sources of entailment rules.

References

1. Dagan, I., Glickman, O.: 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)
2. Dagan, I., Glickman, O., Magnini, B.: The PASCAL RTE challenge. In: PASCAL Challenges Workshop, Southampton, U.K (2005)
3. Bar Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., Szpektor, I.: The II PASCAL RTE challenge. In: PASCAL Challenges Workshop, Venice, Italy (2006)
4. Zanzotto, F.M., Moschitti, A.: 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, Sydney, Australia, Association for Computational Linguistics (July 2006) 401–408

5. Corley, C., Mihalcea, R.: Measuring the semantic similarity of texts. In: Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor, Michigan, Association for Computational Linguistics (June 2005) 13–18
6. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. In: Proc. of the 10th ROCLING, Tapei, Taiwan (1997) 132–139
7. Collins, M., Duffy, N.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: Proceedings of ACL02. (2002)
8. Peñas, A., Rodrigo, A., Sama, V., Verdejo, F.: Overview of the answer validation exercise 2006. In Peters, C., Clough, P., Gey, F., Karlgren, J., Magnini, B., Oard, D., de Rijke, M., Stempfhuber, M., eds.: Evaluation of Multilingual and Multimodal Information Retrieval. LNCS, Heidelberg, Germany, Springer (2007)
9. Charniak, E.: A maximum-entropy-inspired parser. In: Proc. of the 1st NAACL, Seattle, Washington (2000) 132–139
10. Minnen, G., Carroll, J., Pearce, D.: Applied morphological processing of english. Natural Language Engineering **7**(3) (2001) 207–223
11. Miller, G.A.: WordNet: A lexical database for English. Communications of the ACM **38**(11) (November 1995) 39–41
12. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet::similarity - measuring the relatedness of concepts. In: Proc. of 5th NAACL, Boston, MA (2004)
13. Moschitti, A.: Making tree kernels practical for natural language learning. In: Proceedings of EACL'06, Trento, Italy (2006)
14. Joachims, T.: Making large-scale svm learning practical. In Schlkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods-Support Vector Learning, MIT Press (1999)