Recognizing Textual Entailment for Italian EDITS @ EVALITA 2009

Elena Cabrio[1,2], Yashar Mehdad[1,2], Matteo Negri[1], Milen Kouylekov[1], and Bernardo Magnini[1]

[1] Fondazione Bruno Kessler (FBK-irst)
[2] University of Trento
{cabrio,mehdad,negri,kouylekov,magnini}@fbk.eu

Abstract. This paper overviews FBK's participation in the Textual Entailment task at EVALITA 2009. Our runs were obtained through different configurations of EDITS (Edit Distance Textual Entailment Suite), the first freely available open source tool for Recognizing Textual Entailment (RTE). With a 71% Accuracy, EDITS reported the best score out of the 8 submitted runs. We describe the sources of knowledge that have been used (e.g. extraction of rules from Wikipedia), the different algorithms applied (i.e. Token Edit Distance, Tree Edit Distance), and the Particle Swarm Optimization (PSO) module used to estimate the optimal cost of edit operations in the cost scheme. Two different dependency parsers for the annotation of the data in the preprocessing phase have been compared, to assess the impact of the parser on EDITS performances. Finally, the obtained results and error analysis are discussed.

Key words: Textual Entailment, Tree Edit Distance, Open Source System.

1 Introduction

For the first time, the Recognizing Textual Entailment challenge is organized for Italian, as a task of the second evaluation campaign of Natural Language Processing tools for Italian. Taking advantage of past participations to the Pascal RTE Challenge, FBK's submitted runs to EVALITA have been obtained using EDITS (Edit Distance Textual Entailment Suite) [7], an open-source software package for recognizing Textual Entailment developed at FBK. The package, which is freely downloadable at *http://edits.fbk.eu/*[1], provides a basic framework for a distance-based approach to the task, with a highly configurable and customizable environment to experiment with different algorithms. Taking advantage of its potential in terms of extensions and integrations with new algorithms and resources, EDITS was used to run our experiments over the provided datasets.

The paper is structured as follows: Section 2 describes the main features of the EDITS package, its core components, and the workflow. Section 3 presents

[1] The first release of the package, EDITS 1.0, is available under GNU Lesser General Public Licence - LGPL.

the resources we have used as a knowledge source for our EVALITA submissions, while Section 4 describe the settings we have experimented. Section 5 concludes the paper reporting the results we obtained, together with some error analysis.

2 EDITS

The system we have used to take part in the RTE Challenge is the EDITS package (Edit Distance Textual Entailment Suite) [7], developed by the HLT group at FBK. EDITS implements a distance-based approach for recognizing textual entailment, which assumes that the distance between T and H is a characteristic that separates the positive T-H pairs, for which the entailment relation holds, from the negative pairs, for which the entailment relation does not hold (it is developed according to the two way task). More specifically, EDITS is based on edit distance algorithms, and computes the T-H distance as the overall cost of the edit operations (*i.e.* insertion, deletion and substitution) that are necessary to transform T into H.

The edit distance approach used in EDITS builds on three basic components (shown in Figure 1):

- An **Edit distance algorithm**, which calculates the set of edit operations that transform T into H. EDITS provides distance algorithms at three levels: *i) String Edit Distance*, where the three edit operations are defined over sequences of characters, *ii) Token Edit Distance*, where edit operations are defined over sequences of tokens of T and H, and *iii) Tree Edit Distance*, where edit operations are defined over single nodes of a syntactic representation of T and H. EDITS provides an implementation of the Levenshtein distance algorithm [4] for String Edit Distance, a token-based version of the same algorithm for Token Edit Distance, and the Zhang-Shasha algorithm [9] for Tree Edit Distance.
- A **Cost scheme**, which defines the cost associated to each edit operation involving an element of T and an element of H.
- Optional sets of **rules**, both *entailment rules* and *contradiction rules*, providing specific knowledge (*e.g.* lexical, syntactic, semantic) about the allowed transformations between portions of T and H. Each rule has a left hand side (an element of T) and a right hand side (an element of H), associated to a probability which indicates if the left hand side entails or contradicts the right hand side. Rules can be manually defined, or they can be extracted from any external resource available (*e.g.* WordNet, corpora, Wikipedia).

Each module, and its corresponding parameters, can be configured by the user through the XML EDITS Configuration File (ECF), which includes the distance algorithm, the cost scheme, and possible rule repositories. This way EDITS provides a general framework which allows to combine the existing algorithms/cost schemes, or replace them with new ones implemented by the user.

EDITS can work at different levels of complexity, depending on the linguistic analysis carried out over T and H. An internal representation format, called

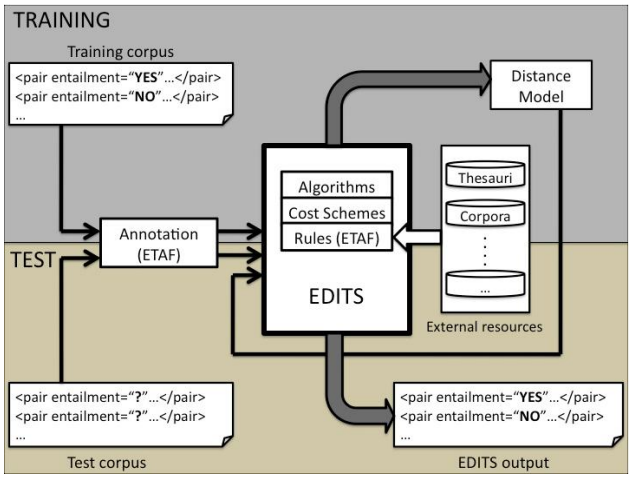

Fig. 1. EDITS workflow.

ETAF (EDITS Text Annotation Format) is defined such that both linguistic processors and semantic resources can be easily used, resulting in a flexible, modular and extensible approach to TE. The format is used both for representing the input (T-H pairs), as well as for representing entailment and contradiction rules. ETAF allows to represent texts at three different levels: *i)* as simple strings; *ii)* as sequences of tokens with their associated morpho-syntactic properties; *iii)* as syntactic trees with structural relations among nodes.

Given a certain configuration of its three basic components, EDITS can be trained over a specific RTE dataset in order to optimize its performance. As shown in Figure 1, in the training phase EDITS produces a *distance model* for the dataset, which consists in a distance threshold S (with $0 < S < K$) that best separates the positive and negative examples in the training data. During the test phase EDITS applies the threshold S, so that pairs resulting in a distance below S are classified as "YES", while pairs above S are classified as "NO". Given the edit distance $ED(T, H)$ for a T-H pair, a normalized entailment score is finally calculated by EDITS using the following formula:

$$entailment(T, H) = \frac{ED(T, H)}{(ED(T, \_) + ED(\_, H))} \quad (1)$$

where $ED(T, H)$ is the function that calculates the edit distance between T and H, and $(ED(T, \_) + ED(\_, H))$ is the distance equivalent to the cost of inserting the entire text of H and deleting the entire text of T. The entailment score has a range from 0 (when T is identical to H), to 1 (when T is completely different from H).

Once a distance model is available, EDITS can be run over a RTE test set. Besides the entailment judgment (*i.e.* "YES"/"NO"), for each pair the system provides the entailment score calculated by the algorithm, and the confidence

score of the entailment assignment (*i.e.* the distance between the entailment score and the threshold S calculated at a training stage).

In order to estimate the optimal cost of each edit operation in the cost scheme, as well as being able to weight the costs such as substituting the terms involved in the entailment rules, a stochastic method based on Particle Swarm Optimization (PSO) [5] was implemented as another module in EDITS. Configuring the PSO module, we try to learn the optimal cost for each edit operation in order to improve the prior textual entailment model. The main goal of using this module is to automatically estimate the best possible operation costs on the development set. Beside that, such method allows to investigate the cost values to better understand how different algorithms approach the data in textual entailment. Moreover, taking advantage of automatic estimation of costs, using PSO would allow the user to find the optimal weight of different resources, and measure their respective contribution on specific datasets.

3 Resources used for EVALITA submissions

An important aspect in dealing with the Textual Entailment task is represented by the amount of knowledge required to correctly handle the input T-H pairs. To address this issue, a list of stopwords, and Wikipedia have been used as the main sources of knowledge used for our EVALITA submissions.

A list of the 149 most frequent Italian words has been collected, and they are used to: *i)* prevent assigning high costs to the deletion/insertion of terms that are unlikely to bring relevant information to detect entailment, and *ii)* to avoid substituting these terms with any content word. Stop words are handled at a *cost scheme* level (*i.e.* by assigning the minimal edit operation cost, 0, as a fixed cost for their insertion/deletion).

Furthermore, 133496 lexical entailment rules have been extracted from the Italian Wikipedia as additional source of knowledge. Rule extraction from Wikipedia is motivated by the high coverage of this resource, specifically in dealing with named entities. Although it represents a potentially noisy knowledge resource, experimental results demonstrated the substantial reliability of the (even suboptimal) rules we collected. Rule extraction has been carried out using the jLSI (java Latent Semantic Indexing) tool [2], as reported in [6].

Applying the extracted entailment rules from Wikipedia, we gained a higher coverage as well as a better performance in our entailment framework. It's worth mentioning that all the entailment rules are represented in the ETAF XML format adopted by EDITS, and are publicly available from the EDITS website.

4 Submissions

This section details the settings of our four submitted runs.

Run 1. We applied the Token Edit Distance algorithm to estimate the distance between T and H. As a preprocessing phase, we used the TextPro tagger

[8] to tag each word as token, lemma, pos, WordNet pos (WNPOS) and full morphological analysis. In the cost scheme, the cost of deletion, insertion and substitution of stop-words were set to 0 and we did not allow substitution of stop-words with content words. No additional rules are applied.

Run 2. The Token Edit Distance algorithm has been applied, with the same cost scheme used for run 1. We assigned a weight to the cost of substituting two terms while they match an entailment rule. These weights were automatically estimated using the PSO algorithm [5] for the set of entailment rules extracted from Wikipedia considering the EVALITA datasets. Also the weight of the words that are not present in the entailment rules were optimized automatically using PSO.

Run 3. The Tree Edit Distance algorithm has been applied on the dependency trees of T and H. As a preprocessing phase, we parsed the data with the Italian version of the MaltParser [3]. As in the previous settings, the cost of deletion, insertion and substitution of stop-words were set to 0. Lexical entailment rules extracted from Wikipedia have been integrated in the configuration, using the same criteria described above.

Run 4. Same setting as run 3, but data has been parsed with the XIP (Xerox Incremental Parser), using the grammar developed for Italian [1].

5 Results and discussion

The results we obtained are illustrated in Table 1. The best run was performed using: *i)* token edit distance to estimate the distance between T and H, *ii)* the rules extracted from Wikipedia, and *iii)* estimating the cost of operations using the PSO [5] module in our settings.

Generally stating, the rules we extracted from Wikipedia were efficient in improving our performance in all runs. Moreover, the automatic estimation of optimized costs played an important role in enhancing and stabilizing the accuracy over the datasets, specifically, while the costs were assigned dynamically.

Table 1. Submissions results (accuracy on training and test data)

	Run1	**Run2**	**Run3**	**Run4**
Dev.	0.72	0.725	0.645	0.647
Test	0.71	0.71	0.51	0.50

In order to assess the impact of the parser on EDITS performances, we compared two different dependency parsers: for the third run we used the Italian version of the MaltParser [3] as the dependency parser, while for the forth run we have used XIP (Xerox Incremental Parser) [1]. As can be seen from the results in Table 1 (run 3 and 4), the system performances on the EVALITA datasets do not vary a lot depending on the parser.

In general, the tree edit distance approach had a sharp drop (about 13 %) in the accuracy over the test set. More specifically, the recall of pairs to be judged as NO ENTAILMENT is really low (in run 3, only 37 non entailing pairs were detected out of 200, and in run 4 only 20). This is due to the fact that EDITS learns the threshold to separate the positive from the negative pairs from the training dataset, and then applies it during the test phase. Most of the pairs contained in the EVALITA datasets are composed by very similar Ts and Hs, that differ between them only of one of a couple of words. Analysing the datasets, we noticed that in the training set there are about one hundred pairs that are more various, that contribute to increase the system threshold. In the test set, almost all pairs are really close, so the threshold is overestimated, and all the sums of the distances calculated for each T-H pairs are lower than the threshold, meaning a positive judgement.

As a general remark, it is worth observing that the best results of our submissions have been obtained applying the Token Edit Distance algorithm, while the use of Tree Edit Distance on this dataset seems to increase the probability of mistakes. The unexpected limited (even detrimental, in our case) contribution of syntactic analysis to our results can be partially explained by the fact that test pairs feature a high word overlap, and share similar syntactic structures, thus reducing the room for syntax-based algorithms. This could be a negative effect of the methodology adopted for dataset creation, which was based on pairs taken from the Italian Wikipedia, manually modified (*e.g.* modifying the tense of a verb, as in T: "*...era membro...*", H: "*...é membro...*") to create contrasting texts. More natural dataset creation methodology would probably increase the distance between results achieved by linear and syntax-based approaches rewarding the second ones.

Acknowledgments. We would like to thank CELI - Language & Information Technology, for parsing the datasets with Xip.

References

1. Bolioli, A., Dini, L., Mazzini, G., Testa: CELI's participation to the Evalita dependency grammar evaluation task. To appear in: Proceedings of the EVALITA 2009. Reggio Emilia, Italy (2009)
2. Giuliano, C.: jLSI a tool for latent semantic indexing. Software available at: http://tcc.itc.it/research/textec/tools-resources/jlsi.html (2007)
3. Lavelli, A., Hall, J., Nilsson, J., Nivre, J.: The Malt Parser at the EVALITA 2009 Dependency Parsing Task. To appear in: Proceedings of the EVALITA 2009.Reggio Emilia, Italy (2009)
4. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Doklady Akademii Nauk SSSR, vol. 163, issue 4(1965)
5. Mehdad, Y.: Automatic Cost Estimation for Tree Edit Distance Using Particle Swarm Optimization. In: Proceedings of the ACL-IJCNLP 2009 Conference (2009)
6. Mehdad, Y., Negri, M., Cabrio, E., Kouylekov, M., Magnini, B.: EDITS: an Open Source Framework for Recognizing Textual Entailment. To appear in: Proceedings of TAC 2009 (2009)

7. Negri, M., Kouylekov, M., Magnini, B., Mehdad Y., Cabrio, E.: Towards Extensible Textual Entailment Engines: the EDITS Package. To appear in: Proceedings of AI*IA 2009 - XIth International Conference of the Italian Association for Artificial Intelligence (2009)
8. Pianta, E., Girardi, C., Zanoli, R.: The TextPro tool suite. In: Proceedings of LREC, 6th edition of the Language Resources and Evaluation Conference (2008)
9. Zhang, K., Shasha D.: Fast Algorithm for the Unit Cost Editing Distance Between Trees. Journal of algorithms, vol. 11 (1990)