

EDITS: An Open Source Framework for Recognizing Textual Entailment

Yashar Mehdad^{1,2}, Matteo Negri¹, Elena Cabrio^{1,2},
Milen Kouylekov¹ and Bernardo Magnini¹

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

Abstract

This paper overviews FBK’s participation in the RTE 5 Evaluation Campaign. Our runs, submitted both to the *main* (two-way classification), and to the *pilot* task, were obtained through different configurations of EDITS (Edit Distance Textual Entailment Suite) package, the first freely available open source RTE software. The main sources of knowledge used, the different configurations, and the achieved results are described, together with ablation tests representing a preliminary analysis of the actual contribution of different resources to the RTE task.

1 Introduction

This year, FBK’s submitted runs to the RTE5 challenge have been obtained using EDITS (Edit Distance Textual Entailment Suite) [5], 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 both the RTE5 tasks (*i.e.* the main, and the pilot task).

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 the resources we have used as a knowledge source for our RTE5 submissions, and the procedures to extract lexical entailment rules from them. Section 4 and Section 5 describe the settings we have experimented for our submissions to the main and to the pilot tasks. Section 6 concludes the paper reporting the results we obtained, together with some error analysis.

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

2 EDITS (Edit Distance Textual Entailment Suite)

The system we have used to take part in the RTE Challenge is the EDITS package (Edit Distance Textual Entailment Suite) [5], 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 components (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 EDITS Configuration File (ECF). A basic configuration file includes at least one distance algorithm and one cost scheme, while rule repositories can be optional. EDITS provides a general framework which allows, through the ECF, to combine in different ways 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 *ETAF* (EDITS Text Annotation Format) is defined such that both linguistic processors and semantic resources can be easily used within EDITS, 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 of annotation: *i)* as simple strings; *ii)* as sequences

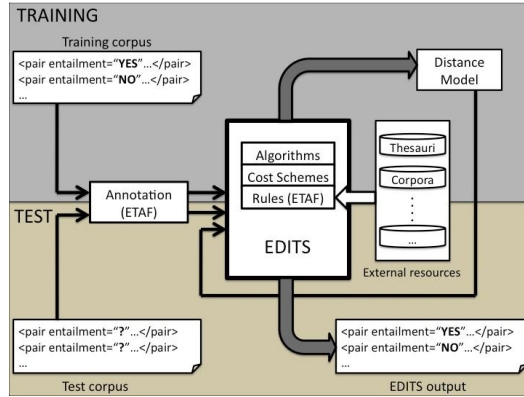


Figure 1: EDITS workflow.

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) [6] 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 RTE5 submissions

An important aspect in dealing with the Textual Entailment problem is represented by the amount of knowledge required to correctly handle the input T-H pairs. To address this issue, the main sources of knowledge used for our RTE5 submissions are: *i)* a list of stopwords, *ii)* WordNet, *iii)* VerbOcean, and *iv)* Wikipedia.

While stop words are simply handled at a *cost scheme* level (*i.e.* by assigning the minimal edit operation cost, 0, as a fixed cost for their insertion/deletion), the knowledge derived from the other resources is handled at the level of *entailment rules*, represented in the ETAF format. Ablation test results reported in Section 6 show the actual contribution of each resource in our experiments with the RTE5 dataset.

As far as **stop words** are concerned, a list of the 572 most frequent English words has been collected. Stop words 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.

WordNet rules. WordNet 3.0 has been used to extract a set of 2698 English entailment rules for terms connected by the *hyponymy* and *synonymy* relations. To reduce the potentially huge amount of extracted rules, the extraction process is *dataset-dependent*, as it considers only the terms in the RTE5 dataset that are connected by the two selected WordNet relations. For instance, if the words "car" and "vehicle" appear in a T/H pair, and *vehicle [has-hyponym] car*, then the rule "car ENTAILS vehicle" is added to the rules repository with a confidence score equal to 1.

Verbocean rules. Verbocean [2] has been used to extract 18232 entailment rules for all the English verbs connected by the "stronger-than" relation. For instance, if "kill [stronger-than] injure", then the rule "kill ENTAILS injure" is added to the rules repository with a confidence score equal to 1. Experimental results demonstrated that computing transitive closures from the first set of verbs directly connected by the *stronger-than* relation introduces noise which is detrimental to the overall system's performance.

Wikipedia rules. Since, the rules extracted from WordNet and VerbOcean often feature limited coverage, XXX lexical entailment rules have been extracted from 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 Wikipedia represents a potentially noisy knowledge resource, experimental results demonstrated the substantial reliability of the (even suboptimal) rules we collected. In contrast with the previous conclusions concerning rules derived from Verbocean, it seems that very precise and accurate rules do not always contribute well in the RTE task.

For this purpose, we computed the Latent Semantic Analysis (LSA) over Wikipedia, as a huge knowledge resource, between all possible node pairs (terms or lemmas) that

appear in the dataset. For this goal we used the jLSI (java Latent Semantic Indexing) tool [11] to measure the relatedness between the term pairs in the dataset. Then, we estimated a relatedness threshold in order to filter all the pairs which has low similarity. In this way, we could arrive to a set of pairs, where the first term entails the second one with a high probability.

The threshold was empirically estimated running a set of experiments. As a result, we found that 0.7, as a relatedness measure, could be a good tradeoff between precision and coverage of the extracted rules. Though a higher threshold could increase precision, leading to more accurate rules, the reduced amount of extracted rules would directly affect coverage, causing an overall performance decrease.

Applying the extracted entailment rules from Wikipedia, we gained a higher coverage as well as a better performance in our entailment framework. As an example, the entailment relations between "Apple" and "Macintosh", or between "Iranian" and "IRIB" could not be captured using WordNet or other resources. However, the mentioned set of rules includes some noise which could be filtered at future.

It's worth mentioning that all our entailment rules are publicly available and were presented in ETAF format, which can be easily converted to any other format.

4 Main Task

For our participation in the main task, we applied the tree edit distance and linear edit distance algorithms over RTE5 dataset. As a preprocessing phase, we used Stanford dependency parser [10] to transform each pair to dependency parse trees. We also applied TextPro tagger [8] in order to prepare our pairs for token edit distance algorithm. Each word was tagged as token, lemma, pos, WordNet pos (WNPOS) and full morphological analysis.

Run 1. In the first run, we applied the Tree Edit Distance algorithm on the parsed trees of text and hypothesis. In the cost scheme, the cost of deletion and insertion and substitution of stop-words were set to 0 and we did not allow substitution of stop-words with content words. In order to estimate the cost of insertion for each token, we divided them into five classes based on their WNPOS. We set the cost of substitution to zero while a lexical entailment rule is applied on a node pair, otherwise we set a dynamic weighted cost for substitution. We integrated the Wikipedia lexical entailment rules which were extracted based on the RTE5 dataset. Finally the cost scheme were optimized using our PSO algorithm [6].

Run 2. In the second run, the Token Edit Distance algorithm has been applied to estimate the distance between text and hypothesis. Same with the first setting, the cost of deletion, insertion and substitution of stop-words were set to 0. In this run, 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 for different sets of entailment rules (Wikipedia and VerbOcean). The weighted entailment rules could support the idea that different source of knowledge might have different effect on the task. Moreover, in this way we reduce the noise of Wikipedia extracted rules vs VerbOcean entailment rules. Furthermore, the weighted dynamic cost of substituting the terms which does not exist in the entailment rules, were optimized automatically.

Run 3. Finally, for the last run, we used the same set of rules and same settings with the second run, however, for each task (IE, IR and QA), one optimized model were obtained. In other words, we obtained three different thresholds based on the task from which each pair has been derived. Consequently, the final decision for each pair of text and hypothesis was dependent on the task annotation of data in the test set. Table 1 summarizes our settings in the three runs we submitted for the main task. It’s worth mentioning that finally, since the rules extracted from Wordnet were covered totally by Wikipedia rules, we decided not to use the WordNet rules in RTE5.

	Algorithms		Rules		Cost Scheme		Optimization		Resources
	Tree	Token	Wiki	VO	Dynamic	Static	PSO	Task	Stop words
Run1	X		X		X		X		X
Run2		X	X	X	X		X		X
Run3		X	X	X	X		X	X	X

Table 1: Main task settings (two way submissions)

5 Pilot Task

In the pilot search task we used EDITS without any peculiar pre- or post- processing module, a clear evidence of the flexibility of our framework in dealing with many real world NLP tasks. The main challenge in the pilot task is to face the problem of having a very unbalanced dataset, with a number of non entailing pairs that is almost ten times larger than the entailing pairs. Most of the current approaches, in particular machine learning ones, are not capable of efficient learning from such a dataset.

To cope with this problem, we configured EDITS so that the threshold is estimated taking into account both the precision and the recall of the entailing pairs simultaneously. This way, the system sets a threshold in the model it builds which increase the F-measure of YES pairs over the whole dataset. Besides that, we configured the swarm optimization module to estimate the optimal costs using F-measure of entailing pairs as the fitness function. The underlying intuition is that driving our current system toward both these directions could be useful in achieving good results in this challenging task.

In the linguistic preprocessing phase, all pairs created by pairing the Ts and the Hs of the documents of the same topic have been merged in a unique dataset, annotated with different tools as described earlier, and represented in the ETAF format.

Run 1. In the first run we submitted, the dataset has been parsed using XIP (Xerox Incremental Parser) [1]. Using the coreference module internal to the parser, both the intrasentential and intersentential coreferent terms have been annotated. To be more precise, all the coreferences detected among terms belonging to the documents of the same topic have been annotated. The tree edit distance algorithm has been used and rules extracted from Wikipedia have been applied. EDITS has been trained on all the possible pairings of Ts and Hs of the documents of the same topic. Among all the models provided by the system (one for each topic), the model whose performances were better (Topic 3) has been applied as a threshold on the test set.

Run 2. In the second run, we applied the token edit distance over the whole dataset previously annotated with TextPro [8]. The costs of insertion, deletion and substitution of stop-words and the cost of substitution of equal tokens were set to 0. The costs of substitution, insertion and deletion of content words were automatically estimated in order to increase the F-measure of entailing pairs. Furthermore, the two sets of lexical rules extracted from Wikipedia and VerbOcean were integrated into EDITS configuration and the weighted cost for substitution of terms matching the rules was automatically estimated using PSO algorithm.

Run 3. In the third run, the model was obtained training EDITS on the whole dataset without using the entailment rules. All other settings and procedures were exactly the same as for the second run.

Table 2 summarizes the setting of the runs we submitted to the pilot search task.

	Algorithms		Rules		Training	Optimization		Resources	
	Tree	Token	Wiki	VO	Topic	PSO	Metric	Stop-w.	Coref
Run1	X	-	X	-	03	X	F1(YES)	X	X
Run2	-	X	X	X	All	X	F1(YES)	X	-
Run3	-	X	-	-	All	X	F1(YES)	X	-

Table 2: Pilot task settings

6 Submission results

We are presenting our results in four main subsection as follows. At first, we explain the results we gained for the main task in two-way classification. Then, we describe our runs and the detailed results of each run in the search pilot task. Furthermore, we show how different resources can affect the task in our system through the ablation test. Finally, we analyze the some errors occurred in our system with the discussion and possible improvements.

6.1 Main Task

Based on the configuration of each run, the results are illustrated in table 3. The best run was performed using tree edit distance over the dependency parse trees of each pair, using the rules extracted from Wikipedia by estimating the cost of operations using PSO [7] module in our settings. It worths mentioning that the highest results, using development set only, were gained over the setting of the third run, while this setting performed the lowest on the test set.

In general for the main task, the linear distance approach had a sharp drop (about 9 %) in the accuracy over the test set, while it gained the highest accuracy on the development set (65.5 %). The main reason can be explained by overfitting the development set using the token edit distance approach. However, based on our assumption, this can be due to the distribution of the development and test set, which needs to be explored and investigated more.

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

	Dev.	Main		QA		IR		IE	
	Acc.	Acc.	Prc.	Acc.	Prc.	Acc.	Prc.	Acc.	Prc.
Run1	0.626	0.602	0.596	0.53	0.566	0.735	0.735	0.54	0.51
Run2	0.632	0.563	0.559	0.525	0.525	0.71	0.732	0.455	0.47
Run3	0.655	0.568	0.573	0.52	0.54	0.705	0.718	0.48	.496

Table 3: Main task results (two-way submission)

6.2 Pilot Task

Table 4 illustrates the results of our submissions for the pilot task based on the settings described in section 5. As we mentioned earlier, the main challenge in the pilot search task was due to the unbalance distribution of entailment pairs with the pairs in which there is no entailment relation. Using our system, estimating the distance threshold using F-measure of entailed pairs, as well as optimizing the costs to move to a better performance, we could achieve a competitive results with no specific pre or post processing.

	Micro Average			Macro avg / topic			Macro avg / hypo		
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
Run1	0.245	0.465	0.321	0.266	0.443	0.332	0.355	0.498	0.415
Run2	0.225	0.648	0.334	0.274	0.635	0.383	0.333	0.680	0.447
Run3	0.218	0.648	0.326	0.275	0.637	0.384	0.330	0.678	0.444

Table 4: Pilot task results

As the results suggest, our best run was performed by estimating the threshold using token edit distance algorithm, optimizing on F-measure of entailed pairs, trained on the whole dataset. Moreover, comparing the results of the second and third run, we can arrive to the conclusion that the lexical entailment rules extracted from Wikipedia played a role in improving the results.

6.3 Ablation tests

To measure the effectiveness of some modules and resources, we tried to ablate them from the first run with the same settings. The results of our ablation tests is illustrated in Table 5. The ablated modules and resources are: 1)the list of stop-words; 2)entailment rules extracted from VerbOcean; 3)entailment rules extracted from Wikipedia and 4)automatic cost estimation using PSO algorithm.

The results indicate that automatic cost estimation has a very high impact on the performance. Besides that, Wikipedia rules and stop-words are effective resources in

Table 5: Ablation tests results.

	Main	Abl. 1		Abl. 2		Abl. 3		Abl. 4	
	Acc.	Acc.	rel.	Acc.	rel.	Acc.	rel.	Acc.	rel.
Run1	60.2%	58.7%	-1.5%	60.3%	+0.1%	59.16%	-1%	57.3%	-3%

recognizing textual entailment using our settings for the first run. Furthermore, another interesting observation reveals that, in contrast with our intuition (and with the results we obtained on the development set), removing the VerbOcean entailment rules slightly rises the results. In order to generalize the statement further tests and investigations must be performed.

6.4 Discussion

Carrying out the error analysis on the output of the system, we noticed that the main problems of our approach can be found in situations of syntactic misalignment of constituents in T and H. Even if using the tree edit distance algorithm on the dependency trees of T and H should help us to deal with such cases, at the moment we are not able to fully exploit the advantages of this kind of representation (e.g., for active/passive alternation or for genitive constructions like *John's brother* \Rightarrow *the brother of John*). For example, given pair 187 in Figure 2, the algorithm is not able to detect the entailment relation between *T:[...] are mostly made up of mangrove trees* and *H: Mangroves are a kind of tree* because of a syntactic misalignment, that brings the system to delete the nodes of T and insert new ones in H even if the semantics of the constituents is the same.

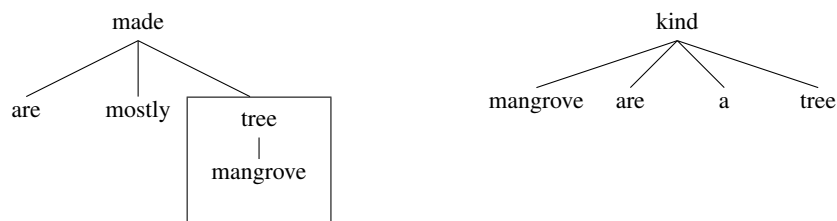


Figure 2: Subtrees extracted from pair 187

This is mainly due to the rigidity of the algorithm. Within a tree, the subtrees are ordered from left to right according to the alphabetical order of the edges linking the nodes to their parents. In the implementation of the tree edit distance algorithm EDITS uses [9], the trees to be compared are numbered using a postorder traversal, meaning that it visits the nodes of the tree starting with the leftmost leaf descendant of the root and proceeding to the leftmost descendant of the right sibling of that leaf, the right sibling(s), then the parent of the leaf and so on up the tree to the root. It means that once the dependency trees are created, the structure is somehow fixed and the algorithm applies on it in a mechanical way.

Furthermore, the algorithms implemented in this version of EDITS do not allow us to apply the edit operations on subtrees, or to take advantage of syntactic rules (e.g. x

verb_active $y \Rightarrow y$ *verb_passive* by x). We are currently working on implementing a new generation of more flexible algorithms in order to deal with this challenge which will be available in the future releases of EDITS.

References

- [1] Ait-Mokhtar, S., Chanod, J.P. and Roux, C. (2002). Robustness beyond shallowness: incremental deep parsing. in *Natural Language Engineering* vol. 8, numbers 2-3, pp. 121–144, Cambridge University Press.
- [2] Chklovski, T. and Pantel, P. (2005). VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)* Barcelona, Spain.
- [3] Kouylekov, M., and Magnini, B. (2005). Tree Edit Distance for Recognizing Textual Entailment. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)* Borovets, Bulgaria.
- [4] Levenshtein, V.I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. In *Doklady Akademii Nauk SSSR*, 163(4).
- [5] Negri, M., Kouylekov, M., Magnini, B., Mehdad Y., Cabrio, E. (2009). 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* Reggio Emilia, Italy, 9-12 December.
- [6] Mehdad, Y. (2009). Automatic Cost Estimation for Tree Edit Distance Using Particle Swarm Optimization. In *Proceedings of the ACL-IJCNLP 2009 Conference* Singapore, 2009.
- [7] Mehdad, Y and Magnini, B. (2009). Optimizing Textual Entailment Recognition Using Particle Swarm Optimization. In *Proceedings of Workshop on Applied Textual Inference, ACL-IJCNLP 2009* Singapore, 2009.
- [8] Pianta, E., Girardi, C., Zanolli, R. (2008). The TextPro tool suite. In *Proceedings of LREC, 6th edition of the Language Resources and Evaluation Conference*. Marrakech, Morocco, 28-30 May 2008.
- [9] Zhang, K., Shasha D. (1990). Fast Algorithm for the Unit Cost Editing Distance Between Trees. in *Journal of algorithms*, vol 11, December.
- [10] D. Klein and C.D. Manning. (2003) Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15* Cambridge, MA. MIT Press.
- [11] Giuliano, C. (2007) jLSI a tool for latent semantic indexing. Software available at <http://tcc.itc.it/research/textec/tools-resources/jLSI.html>.