Is it Worth Submitting this Run?
Assess your RTE System with a Good Sparring Partner

Milen Kouylekov
CELI s.r.l.
Turin, Italy
kouylekov@celi.it

Yashar Mehdad
FBK-irst and University of Trento
Trento, Italy
mehdad@fbk.eu

Matteo Negri
FBK-irst
Trento, Italy
negri@fbk.eu

Abstract

We address two issues related to the development of systems for Recognizing Textual Entailment. The first is the impossibility to capitalize on lessons learned over the different datasets available, due to the changing nature of traditional RTE evaluation settings. The second is the lack of simple ways to assess the results achieved by our system on a given training corpus, and figure out its real potential on unseen test data. Our contribution is the extension of an open-source RTE package with an automatic way to explore the large search space of possible configurations, in order to select the most promising one over a given dataset. From the developers' point of view, the efficiency and ease of use of the system, together with the good results achieved on all previous RTE datasets, represent a useful support, providing an immediate term of comparison to position the results of their approach.

1 Introduction

Research on textual entailment (TE) has received a strong boost by the Recognizing Textual Entailment (RTE) Challenges, organized yearly to gather the community around a shared evaluation framework. Within such framework, besides the intrinsic difficulties of the task (*i.e.* deciding, given a set of *Text-Hypothesis* pairs, if the hypotheses can be inferred from the meaning of the texts), the development of RTE systems has to confront with a number of additional problems and uncertainty factors. First of all, since RTE systems are usually based on complex architectures that integrate a variety of tools and resources, it is *per se* very difficult to tune them and define the optimal configuration given a new dataset. In general, when participating to the evaluation challenges there's no warranty that the submitted runs are those obtained with the best possible configuration allowed by the system. Second, the evaluation settings change along the years. Variations in the length of the texts, the origin of the pairs, the balance between positive and negative examples, and the type of entailment decisions allowed, reflect the need to move from easier and more artificial settings to more complex and natural ones. However, in contrast with other more stable tasks in terms of evaluation settings and metrics (*e.g.* machine translation), such changes make it difficult to capitalize on the experience obtained by participants throughout the years. Third, looking at RTE-related literature and the outcomes of the six campaigns organised so far, the conclusions that can be drawn are often controversial. For instance, it is not clear whether the availability of larger amounts of training data correlates with better performance (Hickl et al., 2006) or not (Zanzotto et al., 2007; Hickl and Bensley, 2007), even within the same evaluation setting. In addition, ablation tests carried out in recent editions of the challenge do not allow for definite conclusions about the actual usefulness of tools and resources, even the most popular ones (Bentivogli et al., 2009). Finally, the best performing systems often have different natures from one year to another, showing alternations of deep (Hickl and Bensley, 2007; Tatu and Moldovan, 2007) and shallow approaches (Jia et al., 2010) ranked at the top positions. In light of these considerations, it would be useful for sys-

Proceedings of the TextInfer 2011 Workshop on Textual Entailment, EMNLP 2011, pages 30–34,
Edinburgh, Scotland, UK, July 30, 2011.

tems developers to have: *i)* automatic ways to support systems' tuning at a training stage, and *ii)* reliable terms of comparison to validate their hypotheses, and position the results of their work before submitting runs for evaluation. In this paper we address these needs by extending an open-source RTE package (EDITS[1]) with a mechanism that automatizes the selection of the most promising configuration over a training dataset. We prove the effectiveness of such extension showing that it allows not only to achieve good performance on all the available RTE Challenge datasets, but also to improve the official results, achieved with the same system, through *ad hoc* configurations manually defined by the developers team. Our contribution is twofold. On one side, in the spirit of the collaborative nature of open source projects, we extend an existing tool with a useful functionality that was still missing. On the other side, we provide a good "sparring partner" for system developers, to be used as a fast and free term of comparison to position the results of their work.

2 "Coping" with configurability

EDITS (Kouylekov and Negri, 2010) is an open source RTE package, which offers a modular, flexible, and adaptable working environment to experiment with the RTE task over different datasets. The package allows to: *i)* create an entailment engine by defining its basic components (i.e. algorithms, cost schemes, rules, and optimizers); *ii)* train such entailment engine over an annotated RTE corpus to learn a model; and *iii)* use the entailment engine and the model to assign an entailment judgement and a confidence score to each pair of an un-annotated test corpus. A key feature of EDITS is represented by its high configurability, allowed by the availability of different algorithms, the possibility to integrate different sets of lexical entailment/contradiction rules, and the variety of parameters for performance optimization (see also Mehdad, 2009). Although configurability is *per se* an important aspect (especially for an open-source and general purpose system), there is another side of the coin. In principle, in order to select the most promising configuration over a given development set, one should exhaustively run a huge number of training/evaluation routines. Such number corresponds to the total number of configurations allowed by the system, which result from the possible combinations of parameter settings. When dealing with enlarging dataset sizes, and the tight time constraints usually posed by the evaluation campaigns, this problem becomes particularly challenging, as developers are hardly able to run exhaustive training/evaluation routines. As recently shown by the EDITS developers team, such situation results in running a limited number of experiments with the most "reasonable" configurations, which consequently might not lead to the optimal solution (Kouylekov et al., 2010).

The need of a mechanism to automatically obtain the most promising solution on one side, and the constraints posed by the evaluation campaigns on the other side, arise the necessity to optimize this procedure. Along this direction, the objective is good a trade-off between exhaustive experimentation with all possible configurations (unfeasible), and educated guessing (unreliable). The remainder of this section tackles this issue introducing an optimization strategy based on genetic algorithms, and describing its adaptation to extend EDITS with the new functionality.

2.1 Genetic algorithm

Genetic algorithms (GA) are well suited to efficiently deal with large search spaces, and have been recently applied with success to a variety of optimization problems and specific NLP tasks (Figueroa and Neumann, 2008; Otto and Riff, 2004; Aycinena et al., 2003). GA are a direct stochastic method for global search and optimization, which mimics natural evolution. To this aim, they work with a *population of individuals*, representing possible solutions to the given task. Traditionally, solutions are represented in binary as strings of *0*s and *1*s, but other encodings (*e.g.* sequences of real values) are possible. The evolution usually starts from a population of randomly generated individuals, and at each generation selects the best-suited individuals based on a *fitness function* (which measures the optimality of the solution obtained by the individual). Such selection is then followed by *modifications* of the selected individuals obtained by recombining (crossover) and performing random changes (mutation) to form a new population, which will be used in the next iter-

[1] http://edits.fbk.eu/

ation. Finally, the algorithm is terminated when the maximum number of generations, or a satisfactory fitness level has been reached for the population.

2.2 EDITS-GA

Our extension to the EDITS package, EDITS-GA, consists in an iterative process that starts with an initial population of randomly generated configurations. After a training phase with the generated configurations, the process is evaluated by means of the fitness function, which is manually defined by the user[2]. This measure is used by the genetic algorithm to iteratively build new populations of configurations, which are trained and evaluated. This process can be seen as the combination of: *i)* a micro training/evaluation routine for each generated configuration of the entailment engine; and *ii)* a macro evolutionary cycle, as illustrated in Figure 1. The fitness function is an important factor for the evaluation and the evolution of the generated configurations, as it drives the evolutionary process by determining the best-suited individuals used to generate new populations. The procedure to estimate and optimize the best configuration applying the GA, can be summarized as follows.

(1) Initialization: generate a random initial population (*i.e.* a set of configurations).

(2) Selection:

2a. The fitness function (accuracy, or F-measure) is evaluated for each individual in the population.

2b. The individuals are selected according to their fitness function value.

(3) Reproduction: generate a new population of configurations from the selected one, through genetic operators (cross-over and mutation).

(4) Iteration: repeat the *Selection* and *Reproduction* until *Termination*.

(5) Termination: end if the maximum number of iterations has been reached, or the population has converged towards a particular solution.

In order to extend EDITS with genetic algorithms, we used a GA implementation available in the JGAP tool[3]. In our settings, each individual contains a sequence of boolean parameters corresponding to the activation/de-activation of the system's basic components (algorithms, cost schemes, rules, and optimizers). The configurations corresponding to such individuals constitute the populations iteratively evaluated by EDITS-GA on a given dataset.

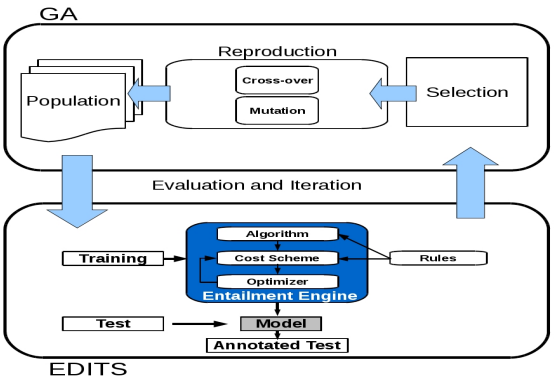

Figure 1: EDITS-GA framework.

3 Experiments

Our experiments were carried out over the datasets used in the six editions of the RTE Challenge ("Main" task data from RTE1 to RTE6). For each dataset we obtained the best model by training EDITS-GA over the development set, and evaluating the resulting model on the test pairs. To this aim, the optimization process is iterated over all the available algorithms in order to select the best combination of parameters. As *termination* criterion, we set to 20 the maximum number of iterations. To increase efficiency, we extended EDITS to pre-process each dataset using the tokenizer and stemmer available in Lucene[4]. This pre-processing phase is automatically activated when the EDITS-GA has to process non-annotated datasets. However, we also annotated the RTE corpora with the Stanford parser plugin (downloadable from the EDITS websitein order to run the syntax-based algorithms available (*e.g.* tree edit distance). The number of boolean parameters used to generate the configurations is 18. In light of this figure, it becomes evident that the number of possible configurations is too large (2^{18}=262,144) for an exhaustive training/evaluation routine over each dataset[5]. However,

[2]For instance, working on the RTE Challenge "Main" task data, the fitness function would be the *accuracy* for RTE1 to RTE5, and the *F-measure* for RTE6.

[3]http://jgap.sourceforge.net/

[4]http://lucene.apache.org/

[5]In an exploratory experiment we measured in around **4 days** the time required to train EDITS, with all possible con-

	# Systems	Best	Lowest	Average	EDITS (rank)	EDITS-GA (rank)	% Impr.	Comp. Time
RTE1	15	0.586	0.495	0.544	0.559 (8)	**0.5787** (3)	+3.52%	8m 24s
RTE2	23	0.7538	0.5288	0.5977	0.605 (6)	**0.6225** (5)	+2.89%	9m 8s
RTE3	26	0.8	0.4963	0.6237	-	**0.6875** (4)	-	9m
RTE4	26	0.746	0.516	0.5935	0.57 (17)	**0.595** (10)	+4.38%	30m 54s
RTE5	20	0.735	0.5	0.6141	0.6017 (14)	**0.6233** (9)	+3.58%	8m 23s
RTE6	18	0.4801	0.116	0.323	0.4471 (4)	**0.4673** (3)	+4.51%	1h 54m 20s

Table 1: RTE results (acc. for RTE1-RTE5, F-meas. for RTE6). For each participant, only the best run is considered.

with an average of 5 *reproductions* on each iteration, EDITS-GA makes an average of 100 configurations for each algorithm. Thanks to EDITS-GA, the average number of evaluated configurations for a single dataset is reduced to around 400[6].

Our results are summarized in Table 1, showing the total number of participating systems in each RTE Challenge, together with the highest, lowest, and average scores they achieved. Moreover, the official results obtained by EDITS are compared with the performance achieved with EDITS-GA on the same data. We can observe that, for all datasets, the results achieved by EDITS-GA significantly improve (up to 4.51%) the official EDITS results. It's also worth mentioning that such scores are always higher than the average ones obtained by participants. This confirms that EDITS-GA can be potentially used by RTE systems developers as a strong term of comparison to assess the capabilities of their own system. Since time is a crucial factor for RTE systems, it is important to remark that EDITS-GA allows to converge on a promising configuration quite efficiently. As can be seen in Table 1, the whole process takes around 9 minutes[7] for the smaller datasets (RTE1 to RTE5), and less than 2 hours for a very large dataset (RTE6). Such time analysis further proves the effectiveness of the extended EDITS-GA framework. For the sake of completeness we gave a look at the differences between the "educated guessing" done by the EDITS developers for the official RTE submissions, and the "optimal" configuration automatically selected by EDITS-GA. Surprisingly, in some cases, even a minor difference in the selected parameters leads to significant gaps in the results. For instance, in RTE6 dataset, the "guessed" configuration (Kouylekov et al., 2010) was based on the lexical overlap algorithm, setting the cost of replacing H terms without an equivalent in T to the minimal Levenshtein distance between such words and any word in T. EDITS-GA estimated, as a more promising solution, a combination of lexical overlap with a different cost scheme (based on the IDF of the terms in T). In addition, in contrast with the "guessed" configuration, stop-words filtering was selected as an option, eventually leading to a 4.51% improvement over the official RTE6 result.

figurations, over small datasets (RTE1 to RTE5).

[6]With these settings, training EDITS-GA over small datasets (RTE1 to RTE5) takes about **9 minutes** each.

[7]All time figures are calculated on an Intel(R) Xeon(R), CPU X3440 @ 2.53GHz, 8 cores with 8 GB RAM.

4 Conclusion

"Is it worth submitting this run?","How *good* is my system?". These are the typical concerns of system developers approaching the submission deadline of an RTE evaluation campaign. We addressed these issues by extending an open-source RTE system with a functionality that allows to select the most promising configuration over an annotated training set. Our contribution provides developers with a good "sparring partner" (a free and immediate term of comparison) to position the results of their approach. Experimental results prove the effectiveness of the proposed extension, showing that it allows to: *i)* achieve good performance on all the available RTE datasets, and *ii)* improve the official results, achieved with the same system, through *ad hoc* configurations manually defined by the developers team.

Acknowledgments

This work has been partially supported by the EC-funded projects CoSyne (FP7-ICT-4-24853), and Galateas (CIP-ICT PSP-2009-3-250430).

References

Margaret Aycinena, Mykel J. Kochenderfer, and David Carl Mulford. 2003. An Evolutionary Approach to Natural Language Grammar Induction. *Stanford CS 224N Natural Language Processing*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, Bernardo Magnini. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. *Proceedings of the TAC 2009 Workshop*.

Ido Dagan and Oren Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. *Proceedings of the PASCAL Workshop of Learning Methods for Text Understanding and Mining*.

Alejandro G. Figueroa and Günter Neumann. 2008. Genetic Algorithms for Data-driven Web Question Answering. *Evolutionary Computation 16(1) (2008) pp. 89-125*.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing Textual Entailment with LCCs Groundhog System. *Proceedings of the Second PASCAL Challenges Workshop*.

Andrew Hickl and Jeremy Bensley. 2007. A discourse commitment-based framework for recognizing textual entailment. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Houping Jia, Xiaojiang Huang, Tengfei Ma, Xiaojun Wan, and Jianguo Xiao. 2010. PKUTM Participation at TAC 2010 RTE and Summarization Track. *Proceedings of the Sixth Recognizing Textual Entailment Challenge*.

Milen Kouylekov and Matteo Negri. 2010. An Open-source Package for Recognizing Textual Entailment. *Proceedings of ACL 2010 Demo session*.

Milen Kouylekov, Yashar Mehdad, Matteo Negri, and Elena Cabrio. 2010. FBK Participation in RTE6: Main and KBP Validation Task. *Proceedings of the Sixth Recognizing Textual Entailment Challenge*.

Yashar Mehdad 2009. *Automatic Cost Estimation for Tree Edit Distance Using Particle Swarm Optimization.* Proceedings of ACL-IJCNLP 2009.

Eridan Otto and María Cristina Riff 2004. Towards an efficient evolutionary decoding algorithm for statistical machine translation. *LNAI, 2972:438447.*.

Marta Tatu and Dan Moldovan. 2007. COGEX at RTE3. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Fabio Massimo Zanzotto, Marco Pennacchiotti and Alessandro Moschitti. 2007. Shallow Semantics in Fast Textual Entailment Rule Learners. *Proceedings of the Third Recognizing Textual Entailment Challenge*.