Optimizing Textual Entailment Recognition Using Particle Swarm Optimization

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

Bernardo Magnini
FBK - Irst
Trento, Italy
magnini@fbk.eu

Abstract

This paper introduces a new method to improve tree edit distance approach to textual entailment recognition, using particle swarm optimization. Currently, one of the main constraints of recognizing textual entailment using tree edit distance is to tune the cost of edit operations, which is a difficult and challenging task in dealing with the entailment problem and datasets. We tried to estimate the cost of edit operations in tree edit distance algorithm automatically, in order to improve the results for textual entailment. Automatically estimating the optimal values of the cost operations over all RTE development datasets, we proved a significant enhancement in accuracy obtained on the test sets.

1 Introduction

One of the main aspects of natural languages is to express the same meaning in many possible ways, which directly increase the language variability and emerges the complex structure in dealing with human languages. Almost all computational linguistics tasks such as Information Retrieval (IR), Question Answering (QA), Information Extraction (IE), text summarization and Machine Translation (MT) have to cope with this notion. Textual Entailment Recognition was proposed by (Dagan and Glickman, 2004), as a generic task in order to conquer the problem of lexical, syntactic and semantic variabilities in languages.

Textual Entailment can be explained as an association between a coherent text (T) and a language expression, called hypothesis (H) such that entailment function for the pair T-H returns the true value when the meaning of H can be inferred from the meaning of T and false, otherwise.

Amongst the approaches to the problem of textual entailment, some methods utilize the notion of distance between the pair of T and H as the main feature which separates the entailment classes (positive and negative). One of the successful algorithms implemented Tree Edit Distance (TED), based on the syntactic features that are represented in the structured parse tree of each string (Kouylekov and Magnini, 2005). In this method the distance is computed as the cost of the edit operations (insertion, deletion and substitution) that transform the text T into the hypothesis H. Each edit operation has an associated cost and the entailment score is calculated such that the set of operations would lead to the minimum cost.

Generally, the initial cost is assigned to each edit operation empirically, or based on the expert knowledge and experience. These methods emerge a critical problem when the domain, field or application is new and the level of expertise and empirical knowledge is very limited. In dealing with textual entailment, (Kouylekov and Magnini, 2006) tried to experiment different cost values based on various linguistics knowledge and probabilistics estimations. For instance, they defined the substitution cost as a function of similarity between two nodes, or, for insertion cost, they employed Inverse Document Frequency (IDF) of the inserted node. However, the results could not proven to be optimal.

Other approaches towards estimating the cost of operations in TED tried to learn a generic or discriminative probabilistic model (Bernard et al., 2008; Neuhaus and Bunke, 2004) from the data, without concerning the optimal value of each operation. One of the drawbacks of those approaches is that the cost values of edit operations are hidden behind the probabilistic model. Additionally, the cost can not be weighted or varied according to the tree context and node location (Bernard et al., 2008).

In order to overcome these drawbacks, we are proposing a stochastic method based on Particle

Proceedings of the 2009 Workshop on Applied Textual Inference, ACL-IJCNLP 2009, pages 36–43,
Suntec, Singapore, 6 August 2009.

Swarm Optimization (PSO), to estimate the cost of each edit operation for textual entailment problem. Implementing PSO, we try to learn the optimal cost for each operation in order to improve the prior textual entailment model. In this paper, the goal is to automatically estimate the best possible operation costs on the development set. A further advantage of such method, besides automatic learning of the operation costs, is being able to investigate the cost values to better understand how TED approaches the data in textual entailment.

The rest of the paper is organized as follows: After describing the TED approach to textual entailment in the next section, PSO optimization algorithm and our method in applying it to the problem are explained in sections 4 and 5. Then we present our experimental setup as well as the results, in detail. Finally, in the conclusion, the main advantages of our approach are reviewed and further developments are proposed accordingly.

2 Tree Edit Distance and Textual Entailment

One of the approaches to textual entailment is based on the Tree Edit Distance (TED) between T and H. The tree edit distance measure is a similarity metric for rooted ordered trees. This metric was initiated by (Tai, 1979) as a generalization of the string edit distance problem and was improved by (Zhang and Shasha, 1989) and (Klein, 1998).

The distance is computed as the cost of editing operations (i.e. insertion, deletion and substitution), which are required to transform the text T into the hypothesis H, while each edit operation on two text fragments A and B (denoted as $A \rightarrow B$) has an associated cost (denoted as γ ($A \rightarrow$ B)). In textual entailment context, the edit operations are defined in the following way based on the dependency parse tree of T and H:

- Insertion ($\lambda \rightarrow A$): insert a node A from the dependency tree of H into the dependency tree of T. When a node is inserted it is attached to the dependency relation of the source label.

- Deletion ($A \rightarrow \lambda$): delete a node A from the dependency tree of T. When A is deleted all its children are attached to the parent of A. It is not required to explicitly delete the children of A, as they are going to be either deleted or substituted in a following step.

- Substitution ($A \rightarrow B$): change the label of a node A in the source tree into a label of a node B of the target tree. In the case of substitution, the relation attached to the substituted node is changed with the relation of the new node.

According to (Zhang and Shasha, 1989), the minimum cost mappings of all the descendants of each node has to be computed before the node is encountered, so the least-cost mapping can be selected right away. To accomplish this the algorithm keeps track of the keyroots of the tree, which are defined as a set that contains the root of the tree plus all nodes which have a left sibling. This problem can be easily solved using recursive methods (Selkow, 1977), or as it was suggested in (Zhang and Shasha, 1989) by dynamic programming. (Zhang and Shasha, 1989) defined the relevant subproblems of tree T as the prefixes of all special subforests rooted in the keyroots. This approach computes the TED (δ) by the following equations:

$$\delta(F_T, \theta) = \delta(F_T - r_{F_T}, \theta) + \gamma(r_{F_T} \rightarrow \lambda) \quad (1)$$

$$\delta(\theta, F_H) = \delta(\theta, F_H - r_{F_H}) + \gamma(\lambda \rightarrow r_{F_H}) \quad (2)$$

$$\delta(F_T, F_H) = min \begin{cases} \delta(F_T - r_{F_T}, F_H) + \gamma(r_{F_T} \rightarrow \lambda) \\ \delta(F_T, F_H - r_{F_H}) + \gamma(\lambda \rightarrow r_{F_H}) \\ \delta(F_T(r_{F_T}), F_H(r_{F_H})) + \\ \delta(F_T - T(r_{F_T}), F_H - H(r_{F_H})) + \\ \gamma(r_{F_T} \rightarrow r_{F_H}) \end{cases} \quad (3)$$

where F_T and F_H are forests of T and H, while r_{F_T} and r_{F_H} are the rightmost roots of the trees in F_T and F_H respectively. θ is an empty forest. Moreover, $F_T(r_{F_T})$ and $F_H(r_{F_H})$ are the forests rooted in r_{F_T} and r_{F_H} respectively.

Estimating δ as the bottom line of the computation is directly related to the cost of each operation. Moreover, the cost of edit operations can simply change the way that a tree is transformed to another. As Figure 1[1] shows (Demaine et al., 2007), there could exist more than one edit script for transforming each tree to another. Based on the

[1]The example adapted from (Demaine et al., 2007)

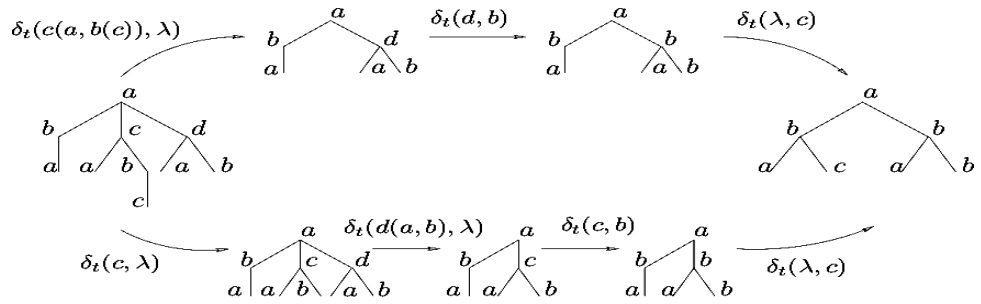

Figure 1: Two possible edit scripts to transform one tree to another.

main definition of this approach, TED is the cost of minimum cost edit script between two trees.

The entailment score for a pair is calculated on the minimal set of edit operations that transform the dependency parse tree of T into H. An entailment relation is assigned to a T-H pair where the overall cost of the transformations is below a certain threshold. The threshold, which corresponds to tree edit distace, is empirically estimated over the dataset. This method was implemented by (Kouylekov and Magnini, 2005), based on the algorithm by (Zhang and Shasha, 1989).

In this method, a cost value is assigned to each operation initially, and the distance is computed based on the initial cost values. Considering that the distance can vary in different datasets, converging to an optimal set of values for operations is almost empirically impossible. In the following sections, we propose a method for estimating the optimum set of values for operation costs in TED algorithm dealing with textual entailment problem. Our method is built on adapting PSO optimization approach as a search process to automate the procedure of the cost estimation.

3 Particle Swarm Optimization

PSO is a stochastic optimization technique which was introduced based on the social behaviour of bird flocking and fish schooling (Eberhart et al., 2001). It is one of the population-based search methods which takes advantage of the concept of social sharing of information. The main structure of this algorithm is not very different from other evolutionary techniques such as Genetic Algorithms (GA); however, the easy implementation and less complexity of PSO, as two main characteristics, are good motivations to apply this optimization approach in many areas.

In this algorithm each *particle* can learn from the experience of other particles in the same population (called *swarm*). In other words, each particle in the iterative search process, would adjust its flying velocity as well as position not only based on its own acquaintance, but also other particles' flying experience in the swarm. This algorithm has found efficient in solving a number of engineering problems. In the following, we briefly explain the main concepts of PSO.

To be concise, for each particle at each iteration, the position X_i (Equation 4) and velocity V_i (Equation 5) is updated. X_{bi} is the best position of the particle during its past routes and X_{gi} is the best global position over all routes travelled by the particles of the swarm. r_1 and r_2 are random variables drawn from a uniform distribution in the range [0,1], while c_1 and c_2 are two acceleration constants regulating the relative velocities with respect to the best local and global positions. The weight ω is used as a tradeoff between the global and local best positions and its value is usually selected slightly less than 1 for better global exploration (Melgani and Bazi, 2008). The optimal position is computed based on the fitness function defined in association with the related problem. Both position and velocity are updated during the iterations until convergence is reached or iterations attain the maximum number defined by the user. This search process returns the best fitness function over the particles, which is defined as the optimized solution.

$$X_i = X_i + V_i \tag{4}$$

$$V_i = \omega V_i + c_1 r_1 (X_{bi} - X_i) + c_2 r_2 (X_{gi} - X_i) \tag{5}$$

Algorithm 1 shows a simple pseudo code of how this optimization algorithm works. In the rest of the paper, we describe our method to integrate this algorithm with TED.

Algorithm 1 PSO algorithm

for all $particles$ **do**
  $Initialize\ particle$
end for
while $Convergence\ or\ maximum\ iteration$ **do**
  for all $particles$ **do**
    $Calculate\ fitness\ function$
    if $fitness\ function\ value > X_{bi}$ **then**
      $X_{bi} \Leftarrow fitness\ function\ value$
    end if
  end for
  $choose\ the\ best\ particle\ amongst\ all\ in\ X_{gi}$
  for all $particles$ **do**
    $calculate\ V_i$
    $update\ X_i$
  end for
end while
return $best\ particle$

4 Automatic Cost Estimation

One of the challenges in applying TED for recognizing textual entailment is estimating the cost of each edit operation which transforms the text T into the hypothesis H in an entailment pair. Since the cost of edit operations can directly affect the distance, which is the main criteria to measure the entailment, it is not trivial to estimate the cost of each operation. Moreover, considering that implying different costs for edit operations can affect the results in different data sets and approaches, it motivates the idea of optimizing the cost values.

4.1 PSO Setup

One of the most important steps in applying PSO is to define a fitness function which could lead the swarm to the optimized particles based on the application and data. The choice of this function is very crucial, since PSO evaluates the quality of each candidate particle for driving the solution space to optimization, on the basis of the fitness function. Moreover, this function should possibly improve the textual entailment recognition model. In order to attain these goals, we tried to define two main fitness functions as follows.

1. **Bhattacharyya Distance**: This measure was proposed by (Bhattacharyya, 1943) as a statistical measure to determine the similarity or distance between two discrete probability distributions. In binary classification, this method is widely used to measure the distance between two different classes. In the studies by (Fukunaga, 1990), Bhattacharyya distance was occluded to be one of the most effective measure specifically for estimating the separability of two classes. Figure 2 shows the intuition behind this measure.

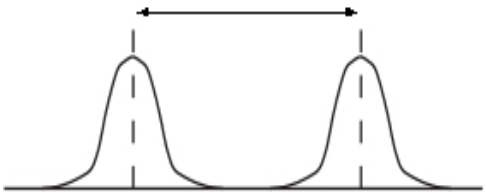

Figure 2: Bhattacharyya distance between two classes with similar variances.

Bhattacharyya distance is calculated based on the covariance (σ) and mean (μ) of each distribution based on its simplest formulation in Equation 6 (Reyes-Aldasoro and Bhalerao, 2006). Maximizing the distance between the classes would result a better separability which aims to a better classification results. Furthermore, estimating the costs using this function would indirectly improve the performance specially in classification problems. It could be stated that, maximizing the Bhattacharyya distance would increase the separability of two entailment classes which result in a better performance.

$$BD(c_1, c_2) = \frac{1}{4} ln\{\frac{1}{4}(\frac{\sigma_{c_1}^2}{\sigma_{c_2}^2} + \frac{\sigma_{c_2}^2}{\sigma_{c_1}^2} + 2)\} + \frac{1}{4}\{\frac{(\mu_{c_1} - \mu_{c_2})^2}{\sigma_{c_1}^2 + \sigma_{c_2}^2}\} \quad (6)$$

2. **Accuracy**: Accuracy or any performance measure obtained from a TED based system, can define a good fitness function in optimizing the cost values. Since maximizing the accuracy would directly increase the performance of the system or enhance the model to solve the problem, this measure is a possible choice to adapt in order to achieve our aim. In this method, trying to maximize the fitness function will compute the best model based on the optimal cost values in the particle space of PSO algorithm.

   In other words, by defining the accuracy obtained from 10 fold cross-validation over the

development set, as the fitness function, we could estimate the optimized cost of the edit operations. Maximizing the accuracy gained in this way, would lead to find the set of edit operation costs which directly increases our accuracy, and consequently guides us to the main goal of optimization.

In the following section, the procedure of estimating the optimal costs are described in detail.

4.2 Integrating TED with PSO for Textual Entailment Problem

The procedure describing the proposed system to optimize and estimate the cost of edit operations in TED applying PSO algorithm is as follows.

a) *Initialization*

Step 1) Generate a random swarm of particles (in a simple case each particle is defined by the cost of three operations).

Step 2) For each position of the particle from the swarm, obtain the fitness function value (Bhattacharyya distance or accuracy) over the training data.

Step 3) Set the best position of each particle with its initial position (X_{bi}).

b) *Search*

Step 4) Detect the best global position (X_{gi}) in the swarm based on maximum value of the fitness function over all explored routes.

Step 5) Update the velocity of each particle (V_i).

Step 6) Update the position of each particle (X_i). In this step, by defining the boundaries, we could stop the particle to exit the allowed search space.

Step 7) For each candidate particle calculate the fitness function (Bhattacharyya distance or accuracy).

Step 8) Update the best position of each particle if the current position has a larger value.

c) *Convergence*

Step 9) Run till the maximum number of iteration (in our case set to 10) is reached or start the search process.

d) *Results*

Step 10) Return the best fitness function value and the best particle. In this step the optimum costs are returned.

Following the steps above, in contrary to determine the entailment relation applying tree edit distance, the operation costs can be automatically estimated and optimized. In this process, both fitness functions could be easily compared and the cost values leading to the better model would be selected. In the following section, the experimental procedure for obtaining the optimal costs by exploiting the PSO approach to TE is described.

5 Experimental Design

In our experiments we show an increase in the performance of TED based approach to textual entailment, by optimizing the cost of edit operations. In the following subsections, the framework and dataset of our experiments are elaborated.

5.1 Dataset Description

Our experiments were conducted on the basis of the Recognizing Textual Entailment (RTE) datasets[2], which were developed under PASCAL RTE challenge. Each RTE dataset includes its own development and test set, however, RTE-4 was released only as a test set and the data from RTE-1 to RTE-3 were used as development set. More details about the RTE datasets are illustrated in Table 5.1.

	Number of pairs			
	Development		Test	
Datasets	YES	NO	YES	NO
RTE-1	283	284	400	400
RTE-2	400	400	400	400
RTE-3	412	388	410	390
RTE-4	—	—	500	500

Table 1: RTE-1 to RTE-4 datasets.

5.2 Experimental Framework

In our experiments, in order to deal with TED approach to textual entailment, we used EDITS[3] package (Edit Distance Textual Entailment Suite)

[2]http://www.pascal-network.org/Challenges/RTE1-4

[3]The EDITS system has been supported by the EU-funded project QALL-ME (FP6 IST-033860). Available at http://edits.fbk.eu/

(Magnini et al., 2009). This system is an open source software based on edit distance algorithms, and computes the T-H distance as the cost of the edit operations (i.e. insertion, deletion and substitution) that are necessary to transform T into H. By defining the edit distance algorithm and a cost scheme (assigning a cost to the edit operations), this package is able to learn a TED threshold, over a set of string pairs, to decide if the entailment exists in a pair.

In addition, we partially exploit the JSwarm-PSO[4] (Cingolani, 2005) package, with some adaptations, as an implementation of PSO algorithm. Each pair in the datasets is converted to two syntactic dependency parse trees using the Stanford statistical parser[5], developed in the Stanford university NLP group by (Klein and Manning, 2003).

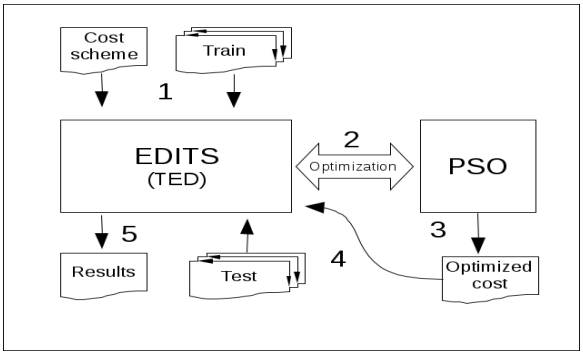

Figure 3: Five main steps of the experimental framework.

In order to take advantage of PSO optimization approach, we integrated EDITS and JSwarm-PSO to provide a flexible framework for the experiments (Figure 5.3). In this way, we applied the defined fitness functions in the integrated system. The Bhattacharyya distance between two classes (YES and NO), in each experiment, could be computed based on the TED score of each pair in the dataset. Moreover, the accuracy, by default, is computed by EDITS over the training set based on 10-fold cross-validation.

5.3 Experimental Scheme

We conducted six different experiments in two sets on each RTE dataset. The costs were estimated on the training set and the results obtained based on the estimated costs over the test set. In the first set of experiments, we set a simple cost scheme based on three operations. Implementing this cost scheme, we expect to optimize the cost of each edit operation without considering that the operation costs may vary based on different characteristics of a node, such as size, location or content. The results were obtained considering three different settings: 1) the random cost assignment; 2) assigning the cost based on the human expertise knowledge and intuition (called Intuitive), and 3) automatic estimated and optimized cost for each operation. In the second case, we used the same scheme which was used in EDITS by its developers (Magnini et al., 2009).

In the second set of experiments, we tried to compose an advanced cost scheme with more fine-grained operations to assign a weight to the edit operations based on the characteristics of the nodes. For example if a node is in the list of stop-words, the deletion cost is set to zero. Otherwise, the cost of deletion would be equal to the number of words in H multiplied by word's length (number of characters). Similarly, the cost of inserting a word w in H is set to 0 if w is a stop word, and to the number of words in T multiplied by words length otherwise. The cost of substituting two words is the Levenshtein distance (i.e. the edit distance calculated at the level of characters) between their lemmas, multiplied by the number of words in T, plus number of words in H. By this intuition, we tried to optimize nine specialized costs for edit operations (i.e. each particle is defined by 9 parameters to be optimized). We conducted the experiments using all three cases mentioned in the simple cost scheme.

In each experiment, we applied both fitness functions in the optimization; however, at the final phase, the costs which led to the maximum results were chosen as the estimated operation costs. In order to save breath and time, we set the number of iterations to 10, in addition, the weight ω was set to 0.95 for better global exploration (Melgani and Bazi, 2008).

6 Results

Our results are summarized in Table 2. We show the accuracy gained by a distance-based (word-overlap) baseline for textual entailment (Mehdad and Magnini, 2009) to be compared with the results achieved by the random, intuitive and optimized cost schemes using EDITS system. For

[4]http://jswarm-pso.sourceforge.net/

[5]http://nlp.stanford.edu/software/lex-parser.shtml

Model		Data set RTE-4	RTE-3	RTE-2	RTE-1
Simple	Random	49.6	53.62	50.37	50.5
	Intuitive	51.3	59.6	56.5	49.8
	Optimized	56.5	61.62	58	58.12
Advanced	Random	53.60	52.0	54.62	53.5
	Intuitive	57.6	59.37	57.75	55.5
	Optimized	59.5	62.4	59.87	58.62
Baseline		55.2	60.9	54.8	51.4
RTE-4 Challenge		57.0			

Table 2: Comparison of accuracy on all RTE datasets based on optimized and unoptimized cost schemes.

the better comparison, we also present the results of the EDITS system in RTE-4 challenge using a combination of different distances as features for classification (Cabrio et al., 2008).

In the first experiment, we estimated the cost of each operation using the simple cost scheme. Table 2 shows that in all datasets, accuracy improved up to 9% by optimizing the cost of each edit operation. Results prove that the optimized cost scheme enhances the quality of the system performance, even more than the cost scheme used by experts (Intuitive cost scheme) (Magnini et al., 2009).

Furthermore, in the second set of experiments, using the fine-grained and weighted cost scheme for edit operations we could achieve the highest results in accuracy. The chart in Figure 4, illustares that all optimized results outperform the word-overlap baseline for textual entailment as well as the accuracy obtained in RTE-4 challenge using combination of different distances as features for classification (Cabrio et al., 2008).

By exploring the estimated optimal cost of each operation, another interesting point was discovered. The estimated cost of deletion in the first set of experiments was 0, which means that deleting a node from the dependency tree of T does not effect the quality of results. This proves that by setting different cost schemes, we could explore even some linguistics phenomena which exists in the entailment dataset. Studying the dataset from this point of view might be interesting to find some hidden information which can not be explored easily.

In addition, the optimized model can reflect more consistency and stability (from 58 to 62 in accuracy) than other models, while in unoptimized models the result varies more, on different datasets (from 50 in RTE-1 to 59 in RTE-3). Moreover, we believe that by changing some parameters such as maximum number of iterations, or by defining a better cost scheme, there could be still a room for improvement.

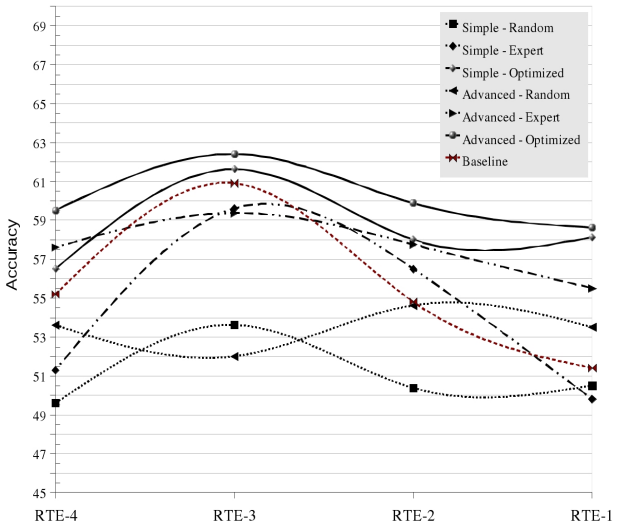

Figure 4: Accuracy obtained by different experimental setups.

7 Conclusion

In this paper, we proposed a novel approach for estimating the cost of edit operations for the tree edit distance approach to textual entailment. With this work we illustrated another step forward in improving the foundation of working with distance-based algorithms for textual entailment. The experimental results confirm our working hypothesis that by improving the results in applying tree edit distance for textual entailment, besides outperforming the distance-based baseline for recog-

nizing textual entailment.

We believe that for further development, extending the cost scheme to find weighted and specialized cost operations to deal with different cases, can lead to more interesting results. Besides that, exploring and studying the estimated cost of operations, could be interesting from a linguistics point of view.

Acknowledgments

Besides my special thanks to Farid Melgani for his helpful ideas, I acknowledge Milen Kouylekov for his academic and technical supports. This work has been partially supported by the three-year project LiveMemories (http://www.livememories.org/), funded by the Provincia Autonoma di Trento.

References

Marc Bernard, Laurent Boyer, Amaury Habrard, and Marc Sebban. 2008. Learning probabilistic models of tree edit distance. *Pattern Recogn.*, 41(8):2611–2629.

A. Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by probability distributions. *Bull. Calcutta Math. Soc.*, 35:99109.

Elena Cabrio, Milen Kouylekovand, and Bernardo Magnini. 2008. Combining specialized entailment engines for rte-4. In *Proceedings of TAC08, 4th PASCAL Challenges Workshop on Recognising Textual Entailment*.

Pablo Cingolani. 2005. Jswarm-pso: Particle swarm optimization package. Available at http://jswarm-pso.sourceforge.net/.

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

E. Demaine, S. Mozes, B. Rossman, and O. Weimann. 2007. An optimal decomposition algorithm for tree edit distance. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 146–157.

Russell C. Eberhart, Yuhui Shi, and James Kennedy. 2001. *Swarm Intelligence*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann.

Keinosuke Fukunaga. 1990. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15*, pages 3–10, Cambridge, MA. MIT Press.

Philip N. Klein. 1998. Computing the edit-distance between unrooted ordered trees. In *ESA '98: Proceedings of the 6th Annual European Symposium on Algorithms*, pages 91–102, London, UK. Springer-Verlag.

Milen Kouylekov and Bernardo Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *PASCAL Challenges on RTE*, pages 17–20.

Milen Kouylekov and Bernardo Magnini. 2006. Tree edit distance for recognizing textual entailment: Estimating the cost of insertion. In *PASCAL RTE-2 Challenge*.

Bernardo Magnini, Milen Kouylekov, and Elena Cabrio. 2009. Edits - edit distance textual entailment suite user manual. Available at http://edits.fbk.eu/.

Yashar Mehdad and Bernardo Magnini. 2009. A word overlap baseline for the recognizing textual entailment task. Available at http://edits.fbk.eu/.

Farid Melgani and Yakoub Bazi. 2008. Classification of electrocardiogram signals with support vector machines and particle swarm optimization. *IEEE Transactions on Information Technology in Biomedicine*, 12(5):667–677.

Michel Neuhaus and Horst Bunke. 2004. A probabilistic approach to learning costs for graph edit distance. In *ICPR '04*, pages 389–393, Washington, DC, USA. IEEE Computer Society.

C. C. Reyes-Aldasoro and A. Bhalerao. 2006. The bhattacharyya space for feature selection and its application to texture segmentation. *Pattern Recogn.*, 39(5):812–826.

Stanley M. Selkow. 1977. The tree-to-tree editing problem. *Inf. Process. Lett.*, 6(6):184–186.

Kuo-Chung Tai. 1979. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433.

K. Zhang and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.