Recognizing Textual Entailment: Experiments with Machine Learning Algorithms and RTE Corpora

Julio J. Castillo
Faculty of Mathematics Astronomy and Physics
Cordoba, Argentina
jotacastillo@gmail.com

Abstract. This paper presents a system that uses machine learning algorithms and a combination of datasets for the task of recognizing textual entailment. The features chosen quantify lexical, syntactic and semantic level matching between text and hypothesis sentences. Additionally, we created a filter which uses a set of heuristics based on Named Entities to detect cases where no entailment was found. We analyze how the different sizes of datasets (RTE1, RTE2, RTE3, RTE4 and RTE5) and classifiers (SVM, AdaBoost, BayesNet, MLP, and Decision Trees) could impact on the final overall performance of the systems. We show that the system performs better than the baseline and the average of the systems from the RTE on both two and three way tasks. We conclude that using RTE3 corpus with Multilayer Perceptron algorithm for both two and three way RTE tasks outperformed any other combination of RTE-s corpus and classifiers in order to predict RTE4 test data.

Keywords: Natural Language Processing, Textual Entailment, Machine Learning, RTE datasets.

1 Introduction

The objective of the Recognizing Textual Entailment Challenge is the task of determining whether or not the meaning of the Hypothesis (H) can be inferred from a text (T). Recently the RTE Challenge has changed to a 3-way task that consists in determining between entailment, contradiction and unknown when there is no information to accept or reject the hypothesis. The traditional two-way distinction between entailment and non-entailment is allowed yet.

In the past RTEs Challenges, machine learning algorithms were widely used for the task of recognizing textual entailment (Marneffe et al., Zanzotto et al., 2007). Thus, in this paper, we tested the most common classifiers that have been used by other researchers in order to provide a common framework of evaluation of ML algorithms (fixing the features) and to show how the development data set could impact over them. We generate a feature vector with the following components for both Text and Hypothesis: Levenshtein distance, a lexical distance based on

Special issue: Natural Language Processing and its Applications.
Research in Computing Science 46, 2010, pp. 155-164

Received 29/11/09
Accepted 16/01/10
Final version 09/03/10

Levenshtein, a semantic similarity measure Wordnet based, and the LCS (longest common substring) metric.

We choose only four features in order to learn the development sets. Larger feature sets do not necessarily lead to improved classification performance because they could increase the risk of overfitting the training data. In section 3 we provide a correlation analysis of these features.

The motivation of the input features: Levenshtein distance is motivated by the good results obtained as measure of similarity between two strings. Additionally, we propose a lexical distance based on Levenshtein distance but working to sentence level. We create a metric based in Wordnet in order to capture the semantic similarity between T and H to sentence level. The longest common substring is selected because is easy to implement and provides a good measure for word overlap.

The system produces feature vectors for all possible combinations of the available development data RTE1, RTE2, RTE3 and RTE5. Weka (Witten and Frank, 2000) is used to train classifiers on these feature vectors. We experiment with the following five machine learning algorithms: Support Vector Machine (SVM), AdaBoost (AB), BayesNet (BN), Multilayer Perceptron(MLP), and Decision Trees(DT). The Decision Trees are interesting because we can see what features were selected for the top levels of the trees. SVM, Bayes Net and AdaBoost were selected because they are known for achieving high performance. MLP was used because has achieved high performance in others NLP tasks.

We experiment with various parameters (settings) for the machine learning algorithms including only the results for the best parameters.

For two-way classification task, we use the RTE1, RTE2, RTE3 development sets from Pascal RTE Challenge, RTE5 gold standard and BPI test suite (Boing, 2008). For three-way task we use the RTE1, RTE2, RTE3 development sets from Stanford group, and RTE5 gold standard set. Additionally, we generate the following development sets: RTE1+RTE2, RTE2+RTE3, RTE1+RTE3, RTE1+RTE5, RTE2+RTE5, RTE3+RTE5, RTE2+RTE3+RTE5, RTE1+RTE2+RTE3 and RTE1+RTE2+RTE3+RTE5 in order to train with different corpus and different sizes. In all cases, RTE4 TAC 2008 gold standard dataset was used as test-set.

The remainder of the paper is organized as follows. Section 2 describes the architecture of our system, whereas Section 3 shows results of experimental evaluation and discussion of them. Finally, Section 4 summarizes the conclusions and lines for future work.

2 System description

This section provides an overview of our system that was evaluated in Fourth Pascal RTE Challenge. The system is based on a machine learning approach for recognizing textual entailment. In Figure 1 we present a brief overview of the system.

Using a machine learning approach we tested with different classifiers in order to classify RTE-4 test pairs in three classes: entailment, contradiction or unknown. To deal with RTE4 in a two-way task, we needed to convert this corpus only in two classes: yes, and no. For this purpose, both contradiction and unknown were taken as class no.

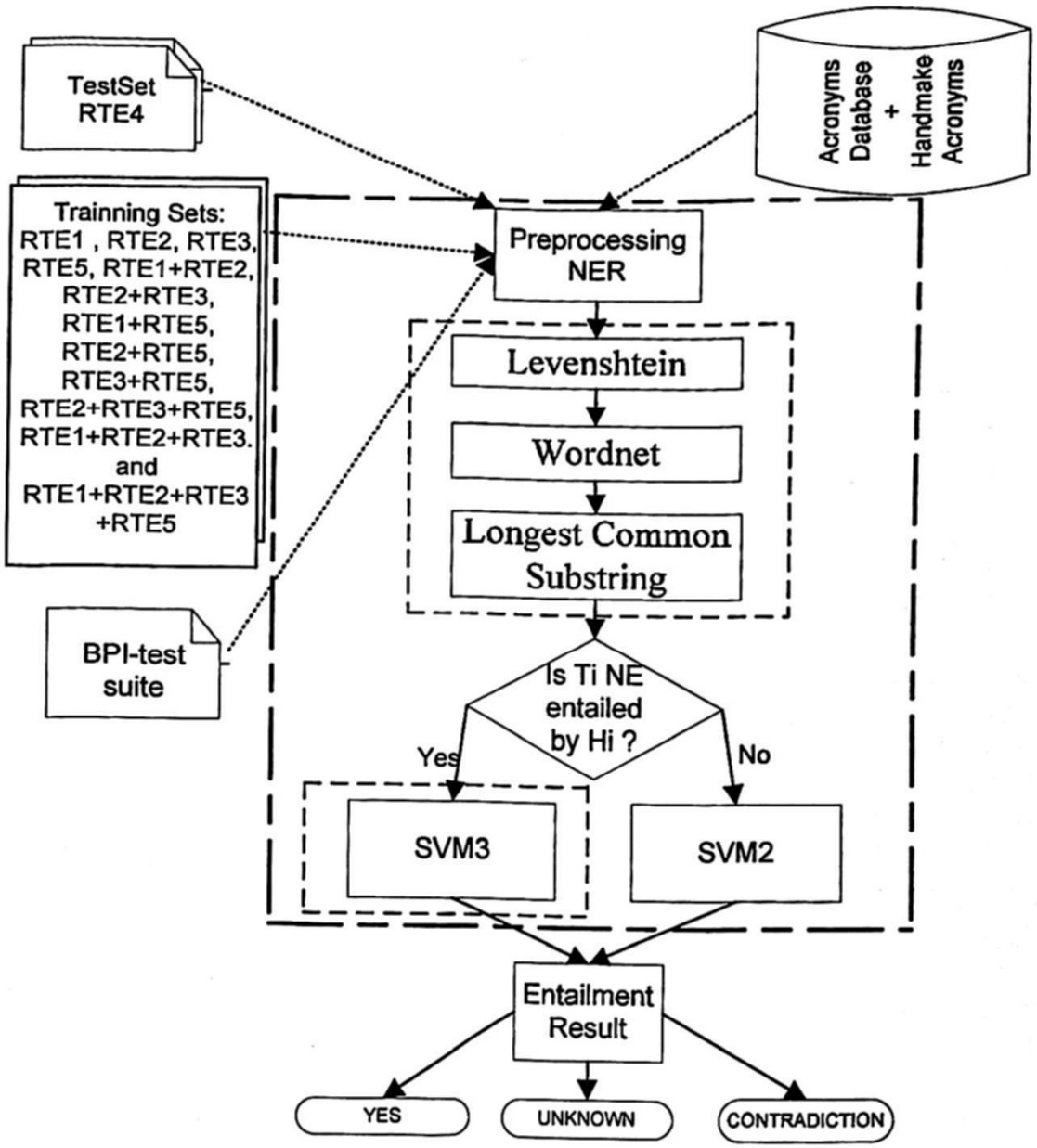

Figure 1. General architecture of our system.

There are two variants to deal with every particular text-hypothesis pair or instance. The first way is directly using four features: (1) the Levenshtein distance between each pair, (2) lexical distance based on Levenshtein, (3) a semantic distance based on WordNet and (4) their Longest Common Substring. The second way is using the "NER- preprocessing module" to determinate whether *non-entailment* is found between text-hypothesis, therefore differing only on the treatment of Named Entities. The Levenshtein distance (Levenshtein, 1966) is computed among the characters in the stemmed Text and Hypothesis strings. The others three features are detailed below.

Text-hypothesis pairs are stemmed with Porter's stemmer and PoS tagged with the tagger in the OpenNLP framework.

2.1 NER Filter

The system applies a filter based on Named Entities. The purpose of the filter is to identify those pairs where the system is sure that *no entailment* relation occurs.

Thus, the NER-preprocessing module performs NER in text-hypothesis pairs applying several heuristics rules to discard when an entailment relation is not found in the pair. In this case, a specialized classifier SVM_2 was trained only with *contradiction* and *unknown* cases of RTE3 corpus and used to classify the pairs between these two classes.

We employed the following set of heuristic rules: for each type of Name Entity (person, organization, location, etc.), if there is a NE of this type occurring in H that does not occur in T, then the pair does not convey an entailment and therefore should be classified as either *contradiction* or *unknown*.

The text-hypothesis pairs are tokenized with the tokenizer of OpenNLP framework and stemmed with Porter's stemmer. We also enhanced this NER-preprocess module by using an acronym database (British Atmospheric Data Centre (BADC)).

The output module was applied to approximately 10 percent of the text-hypothesis pairs of RTE4. The accuracy of the filter evaluated in TAC'08 was 0.71, with 66 cases correctly classified out of 92 where rules applied. An error analysis revealed that misclassified cases were indeed difficult cases, as in the following example (pair 807, RTE4):

Text:
`Large scores of Disney fans had hoped Roy would read the Disneyland Dedication Speech on the theme park's fiftieth birthday next week, which was originally read by Walt on the park's opening day, but Roy had already entered an annual sailing race from Los Angeles to Honolulu.`

Hypothesis:
`Disneyland theme park was built fifty years ago.`

We plan to extend this module so it can also be used to filter cases where an entailment between text and hypothesis can be reliably identified via heuristic rules.

2.2 Lexical Distance

We use the standard Levenshtein distance as a simple measure of how different two text strings are. This distance quantifies the number of changes (character based) to generate one text string from the other. For example, how many changes are necessary in the hypothesis H to obtain the text T. For identical strings, the distance is 0.

Additionally using Levenshtein distance we define a lexical distance and the procedure is the following:

- Each string T and H are divided in a list of tokens.
- The similarity between each pair of tokens in T and H is performed using the Levenshtein distance.

- The string similarity between two lists of tokens is reduced to the problem of "bipartite graph matching", being performed by using the Hungarian algorithm over this bipartite graph. Then, we find the assignment that maximizes the sum of ratings of each token. Note that each graph node is a token of the list.

  Finally the final score is calculated by:

$$finalscore = \frac{TotalSim}{Max(Lenght(T), Lenght(H))}$$

  Where:
  TotalSim is the sum of the similarities with the optimal assignment in the graph.
  Length (T) is the number of tokens in T.
  Length (H) is the number of tokens in H.

2.3 Wordnet Distance

WordNet is used to calculate the semantic similarity between a T and a H. The following procedure is applied:

1. Word sense disambiguation using the Lesk algorithm (Lesk, 1986), based on Wordnet definitions.
2. A semantic similarity matrix between words in T and H is defined. Words are used only in synonym and hyperonym relationship. The Breadth First Search algorithm is used over these tokens; similarity is calculated using two factors: length of the path and orientation of the path.
3. To obtain the final score, we use matching average.

The semantic similarity between two words (step 2) is computed as:

$$Sim(s,t) = 2 \times \frac{Depth(LCS(s,t))}{Depth(s) + Depth(t)}$$

Where:

s,t are source and target words that we are comparing (s is in H and t is in T).
Depth(s) is the shortest distance from the root node to the current node.
LCS(s,t):is the least common subsume of s and t.
The matching average (step 3) between two sentences X and Y is calculated as follows:

$$MatchingAverage = 2 \times \frac{Match(X,Y)}{Length(X) + Length(Y)}$$

2.4 Longest Common Substring

Given two strings, T of length n and H of length m, the Longest Common Sub-string (LCS) method (Dan, 1999) will find the longest strings which are substrings of both T and H. It is founded by dynamic programming.

$$lcs(T,H) = \frac{Length(MaxComSub(T,H))}{\min(Length(T), Length(H))}$$

In all practical cases, min(Length(T), Length(H)) would be equal to Length(H). All values will be numerical in the [0,1] interval.

3 Experimental Evaluation and Discussion of the Results

We use the following combination of datasets: RTE1, RTE2, RTE3, RTE5(gold-set), BPI, RTE1+RTE2, RTE1+RTE3, RTE2+RTE3, RTE1+RTE5, RTE2+RTE5, RTE3+RTE5, RTE1+RTE2+RTE3, and RTE1+RTE2+RTE3+RTE5 to deal with two-way classification task; and we use the following combination of datasets: RTE1[1], RTE2[1], RTE3[1], RTE5, RTE1+RTE2, RTE2+RTE3, RTE1+RTE3, RTE1+RTE5, RTE2+RTE5, RTE3+RTE5, RTE1+RTE2+RTE3, and RTE1+RTE2+RTE3+RTE5 to deal with three-way classification task. We use the following five classifiers to learn every development set: Support Vector Machine, Ada Boost, Bayes Net, Multilayer Perceptron (MLP) and Decision Tree using the open source WEKA Data Mining Software (Witten & Frank, 2005). In all tables results we show only the accuracy of the best classifier. The RTE4 test set and RTE5-gold set were converted to "RTE4 2-way" and "RTE5-2way" taking *contradiction* and *unknown* pairs as no- entailment in order to assess the system in the two-way task. Tables 1 and 2 shows the results for two-way and three-way task, respectively.

Table 1. Results of two-way classification task.

Training Set	Classifier	Acc %
RTE3	MLP	**58.4%**
RTE3 + RTE5	MLP	57.8%
RTE3 With NER Module	SVM	57.6%
RTE2 + RTE3	MLP	57.5%
RTE1 + RTE2 + RTE3	MLP	57.4%
RTE1+RTE5	MLP	57.2%
RTE5	SVM	57.1%
RTE1 + RTE3	Decision Tree	57.1%
RTE1+RTE2+RTE3+RTE5	MLP	57%
RTE2 + RTE3 + RTE5	Bayes Net	56.9%
RTE2 + RTE5	Decision Tree	56.3%
RTE1 + RTE2	Decision tree	56.2%
RTE2	ADA Boost	55.6%
RTE1	ADA Boost	54.6%
Baselines	-	50%
BPI	BayesNet	49.8%

[1] Data set from Stanford Group.

Table 2. Results of three-way classification task.

Training Set	Classifier	Acc %
RTE3	MLP	**55.4%**
RTE2+RTE3+RTE5	MLP	55.3%
RTE1 + RTE3	MLP	55.1%
RTE1 + RTE5	SVM	54.9%
RTE3 + RTE5	SVM	54.9%
RTE1 + RTE2 + RTE3	MLP	54.8%
RTE1 + RTE2	SVM	54.7%
RTE2	SVM	54.6%
RTE2+RTE3	MLP	54.6%
RTE5	SVM	54.6%
RTE1+RTE2+RTE3+RTE5	SVM	54.6%
RTE2 + RTE5	SVM	54.5%
RTE1	SVM	54%
RTE3-With NER Module	SVM	53.8%
Baseline	-	50%

Here we note that, in both classification tasks (two and three way), using RTE3 instead of RTE2 or RTE1 always achieves better results. Interestingly, the RTE3 training set alone outperforms the results obtained with any other combination of RTE-s datasets, even despite the size of increased corpus. Thus, for training purpose, it seems that any additional datasets to RTE-3 introduces "noise" in the classification task.

(Zanzotto et al, 2007) showed that RTE3 alone could produce higher results that training on RTE3 merged with RTE2 for the two-way task. Thus, it seems that it is not always true that more learning examples increase the accuracy of RTE systems. These experiments provide additional evidence for both classification tasks. However, this claim is still under investigation.

Always the RTE1 dataset yields the worst results, maybe because this dataset has been collected with different text processing applications (QA, IE, IR, SUM, PP and MT), and our system does not have it into account.

In addition, a significant difference in performance of 3.8% and 8.6% was obtained using different corpus, in two-way classification task (with and without the BPI development set, respectively).

In three-way task a slight and not statistical significant difference of 1.4% between the best and worst combination of datasets and classifiers is found. So, it suggests that the combination of dataset and classifier has more impact over 2-way task than over 3-way task.

The best performance of our system was achieved with Multilayer Perceptron classifier with RTE-3 dataset; it was 58.4% and 55.4% of accuracy, for two and three way, respectively. The average difference between the best and the worst classifier for all datasets in two way task was 1.6%, and 2.4% in three-way task.

On the other hand, even if the SVM classifier does not appear as 'favorite' in any classification task, in average SVM is one of the best classifiers.

The performance in all cases was clearly above those baselines. Only using BPI in two-way classification we have obtained a result worst that baseline, and it is because, BPI is syntactically simpler than PASCAL RTE; therefore, it seems to be not enough good training set for machine learning algorithm.

Although the best results were obtained without using the Name Entity Preprocessing module, we believe these results could be enhanced. The accuracy of this module was 71%, but additional analysis of the misclassified instances provide evidence that it could be improved almost up to 80% (e.g.: improving the acronym database, knowledge base information, etc) and thus it could impact positively in overall performance of the system.

With the aim of analyzing the feature-dependency, we calculated the correlation of them. The correlation and causation are connected, because correlation is needed for causation to be proved. The correlation matrix of features is shown below:

Table 3. Correlation matrix of features.

Features	1	2	3	4
1	-	0,8611	0,6490	0,2057
2	0,8611	-	0,6951	0,0358
3	0,6490	0,6951	-	0,1707
4	0,2057	0,0358	0,1707	-

The table shows that features (1) and (2) are strongly correlated, so we experimented eliminating feature (1) to assess the effect on the overall performance over cross validation, and we obtained that accuracy slight decreases in 1%. Similar results are obtained eliminating feature (2).

Finally, we assess our system using cross validation technique with ten folds to every corpus, testing over our five classifiers for both classification tasks. The results are shown in the tables 4 and 5 below.

Table 4. Results obtained with ten folds Cross Validation in three-way task.

Training Set	Classifier	Acc %
RTE3	MLP	**65.5%**
RTE3+RTE5	MLP	61.42%
RTE2 + RTE3	MLP	60.68%
RTE1+RTE2+RTE3	MLP	59.35%
RTE2+RTE3+RTE5	SVM	58.72%
RTE1+RTE2+RTE3+RTE5	SVM	57.74%
RTE5	MLP	57.16%
RTE2	SVM	56.62%
RTE1+RTE2	SVM	55.84%
RTE2+RTE5	MLP	55.28%
RTE1	Decision tree	54.70%
RTE1+RTE5	MLP	53.88%

Table 5. Results obtained with Cross Validation in two-way task.

Training Set	Classifier	Acc %
RTE3	BayesNet	**67.85%**
BPI	BayesNet	64%
RTE1 + RTE2 + RTE3	MLP	63.16%
RTE3+RTE5	BayesNet	63.07%
RTE2+RTE3+RTE5	MLP	61.77%
RTE1+RTE2+RTE3+RTE5	ADA Boost	60.91%
RTE5	SVM	60.33%
RTE2	SVM	60.12%
RTE1+RTE2	MLP	59.79%
RTE2+RTE5	BayesNet	58.78%
RTE1	SVM	57.83%
RTE1+RTE5	SVM	56.93%

The results on test set are worse than those obtained on training set, which is most probably due to overfitting of classifiers and because of the possible difference between these datasets. As before, RTE3 outperforms any other combinations of data sets in ten-fold Cross Validation for both two and three way task.

4 Conclusions

We presented our RTE system which is based on a wide range of machine learning classifiers and datasets. As a conclusion about development sets, we mention that the results performed using RTE3 were very similar to those obtained by the union of the RTE1+ RTE2+RTE3 and RTE3 + RTE5, for both 2-way and 3-way tasks. Thus the claim that using more training material helps seems not to be supported by these experiments.

Additionally, we concluded that the relatively similar performances of RTE3 and RTE3 with NER preprocessing module suggest that further refinements over heuristic rules can achieve better results.

Despite the fact that we did not present here an exhaustive comparison between all available datasets and classifiers, we can conclude that the best combination of RTE-s datasets and classifier chosen for two way task produces more impact that the same combination for three way task, almost for all experiments that we have done. In fact, the use of RTE3 alone improves the performance of our system. Thus, we conclude that RTE3 corpus for both two and three way outperforms any other combination of RTE-s corpus using Multilayer Perceptron classifier.

Future work is oriented to experiment with additional lexical and semantic similarities features and to test the improvements they may yield. Additional work will be focus on improving the performance of our NE preprocessing module.

References

1. Prodromos Malakasiotis and Ion Androutsopoulos. *Learning Textual Entailment using SVMs and String Similarity Measures*. ACL-PASCAL, Prague (2007)
2. Julio Javier Castillo, and Laura Alonso. *An approach using Named Entities for Recognizing Textual Entailment*. TAC 2008, Maryland, USA (2008)
3. M. Lesk. *Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone.* In SIGDOC '86 (1986)
4. Gusfield, Dan. Algorithms on Strings, Trees and Sequences. CUP (1999)
5. V. Levenshtein. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals.* Soviet Physics Doklady, 10:707 (1966)
6. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco (2005)
7. D. Inkpen, D. Kipp and V. Nastase. *Machine Learning Experiments for Textual Entailment.* RTE2 Challenge, Venice, Italy (2006)
8. F. Zanzotto, Marco Pennacchiotti and Alessandro Moschitti. *Shallow Semantics in Fast Textual Entailment Rule Learners,* RTE3, Prague (2007)
9. Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty and Christopher D. Manning. *Learning to distinguish valid textual entailments.* RTE2 Challenge, Italy (2006)