A Study of Machine Learning Algorithms for Recognizing Textual Entailment

Julio Javier Castillo

Faculty of Mathematic Astronomy and Physics - National University of Cordoba
Córdoba, Argentina
jotacastillo@gmail.com

Abstract

This paper presents a system that uses machine learning algorithms and a combination of data sets for the task of recognizing textual entailment. The chosen features quantify lexical, syntactic and semantic level by matching between texts 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 analyzed how the different sizes of data sets and classifiers 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 concluded that evaluating using the RTE3 test set, the model learned using MLP from the RTE3 alone outperforms other models that employed different ML algorithms and additional training data from the RTE1 and RTE 2.

Keywords

Textual entailment, machine learning, rte data sets.

1. Approach

The objective of the Recognizing Textual Entailment Challenge is determining whether the meaning of the Hypothesis (H) can be inferred from a text (T). Recently the RTE4 Challenge has changed to a 3-way task that consists in distinguish among entailment, contradiction and unknown when there is no information to accept or reject the hypothesis. However the traditional two-way distinction between entailment and non-entailment is still allowed.

In the past, RTEs Challenges machine learning algorithms were widely used for the task of recognizing textual entailment (Marneffe et al., Zanzotto et al.). 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 showing how the development data set could impact over them.

We generated a feature vector with the following components for both Text and Hypothesis:

- Levenshtein distance,
- Lexical level: a lexical distance based on Levenshtein,
- Semantic level: a semantic similarity measure Wordnet based,
- LCS (longest common substring) metric.

We chose only four features in order to learn the development sets. Larger feature sets do not necessarily lead to improving classification performance because it 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 a measure of similarity between two strings. Additionally, we proposed a lexical distance which is based on Levenshtein distance but working to sentence level.
We created a metric based on Wordnet to try to capture the semantic similarity between T and H to sentence level.
Longest common substring is selected because is easy to implement and provides a good measure for word overlap.
Furthermore, the system uses a NER filter that detects cases where no entailment relation is found. This filter applies heuristic rules over Named Entities found in the text and hypothesis.

The system produces feature vectors for all possible combinations of the available development data RTE1, RTE2 and RTE3. Weka (Witten and Frank, 2000) is used to train classifiers on these feature vectors. We experimented with the following five machine learning algorithms:

- Support Vector Machine (SVM),
- AdaBoost (AB),
- BayesNet (BN),
- Multilayer Perceptron (MLP),
- Decision Trees (DT).

Student Research Workshop, RANLP 2009 - Borovets, Bulgaria, pages 12–17

The Decision Trees are interesting because we can see what features were selected from the top levels of the trees. SVM, Bayes Net and AdaBoost were selected because they are known for achieving high performances. MLP was used because has achieved high performance in others NLP tasks.
We experimented with various parameters (settings) for the machine learning algorithm, such like increasing the confidence factor in DT for more pruning of the trees, different configuration(layers and neurons) for the neural network, and different kernels for SVM. Thus, we tested classifiers used by other researchers in order to provide a common framework of evaluation.

For two-way classification task, we used the RTE1, RTE2, RTE3 development sets from Pascal RTE Challenge, and BPI1 test suite.

For three-way task we used the RTE1, RTE2 and RTE3 development sets from Stanford group2.

Additionally, we generated the following development sets: RTE1+RTE2, RTE2+RTE3, RTE1+RTE3, and RTE1+RTE2+RTE3 in order to train with different corpus and different sizes. In all the cases, RTE4 TAC 2008 gold standard data set 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 the 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 into two classes: yes and no. For this purpose both contradiction and unknown were taken as class *no*.

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 [5] is computed between 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 [3] and PoS tagged with the tagger in the OpenNLP[3] framework.

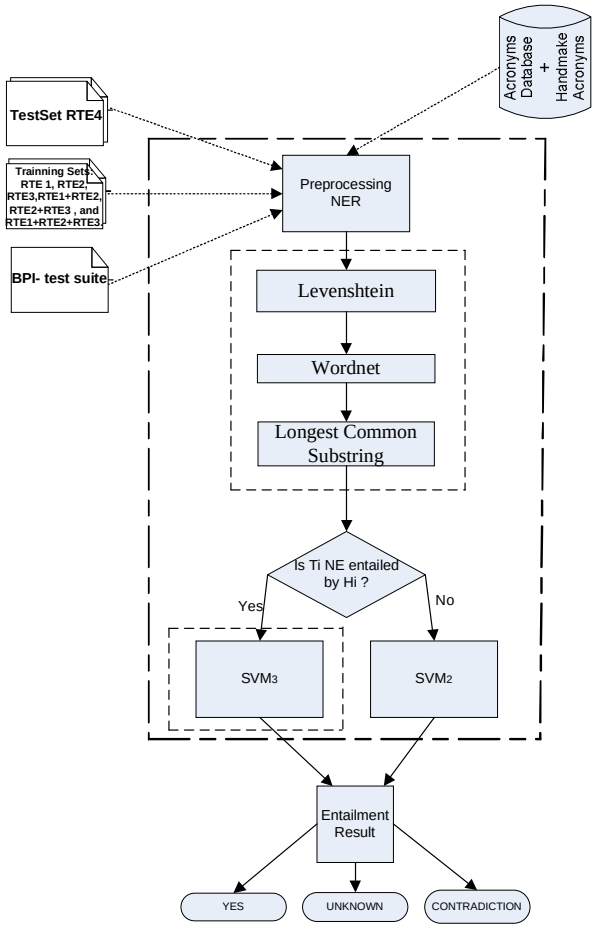

Figure 1.General architecture of our system.

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

[1] http://www.cs.utexas.edu/users/pclark/bpi-test-suite/

[2] http://www-nlp.stanford.edu/projects/contradiction/

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

system is sure that *no entailment* relation occurs, performing a two steps procedure.

Thus, in the first step 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. After that, 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 the 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[4] [3]. We also enhanced this NER-preprocess module by using an acronym database [8].

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: `Larges 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.`

It was misclassified because of the entity date "fifty years ago" is present in H but not in T. The module unknowns that "fifty years ago" refers to the same date event as "fiftieth birthday".

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 (zero).

Additionally, using Levenshtein distance we defined 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", performed using the Hungarian algorithm over this bipartite graph. Then, we found the assignment that maximizes the sum of ratings of each token. Note that each graph node is a token of the list.

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 an H. The following procedure is applied:

1. Word sense disambiguation using the Lesk algorithm [4], 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 by using two factors: length of the path and orientation of the path.

3. To obtain the final score, we use matching average. A bipartite graph is built and computed using Hungarian algorithm.

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

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

[4] http://tartarus.org/~martin/PorterStemmer/

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.

Finally, 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 [5] 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) . Therefore, all values will be numerical in the [0,1] interval.

3. Experimental Evaluation and Discussion of Results

With the aim of exploring the differences between the training sets and machine learning algorithms, we did many experiments looking for the best result to our system.

Thus, we used the following combination of datasets: RTE1, RTE2, RTE3, BPI[5], RTE1+RTE2, RTE1+RTE3, RTE2+RTE3 and RTE1+RTE2+RTE3 to deal with two-way classification task.

In a similar way, we used the following combination of datasets: RTE1, RTE2, RTE3, RTE1+RTE2, RTE2+RTE3, RTE1+RTE3 and RTE1+RTE2+RTE3 of Stanford Group to deal with three-way classification task.

We used five classifiers to learn every development set: (1) Support Vector Machine, (2) Ada Boost, (3) Bayes Net, (4) Multilayer Perceptron (MLP) and (5) Decision Tree using the open source WEKA Data Mining Software [7]. In all the tables results we show only the accuracy of the best classifier.

[5] http://www.cs.utexas.edu/users/pclark/bpi-test-suite/

The RTE4 data set is three-way. Nevertheless, this corpus was converted into "RTE4 2-way" taking *contradiction* and *unknown* pairs as no- entailment in order to test the system in the two-way task.

Our results for RTE two-way classification task are summarized in Table 1 below. In addition, table 2 shows the results obtained in RTE three-way classification task.

Dataset	Classifier	Acc %
RTE3	**MLP**	**58.4%**
RTE3 With NER Module	SVM	57.6%
RTE2 + RTE3	MLP	57.5%
RTE1 + RTE2 + RTE3	MLP	57.4%
RTE1+ RTE3	Decision Tree	57.1%
RTE1 + RTE2	Decision tree	56.2%
RTE2	ADA Boost	55.6%
	Decision tree	55.6%
	Bayes Net	55.6%
RTE1	ADA Boost	54.6%
	Bayes Net	54.6%
Baselines	-	50%
BPI	BayesNet	49.8%

Table 1.Results obtained in two-way classification task.

Dataset	Classifier	Acc %
RTE3	**MLP**	**55.4%**
RTE1 + RTE3	MLP	55.1%
RTE1 + RTE2 + RTE3	MLP	54.8%
RTE1 + RTE2	SVM	54.7%
RTE2	SVM	54.6%
RTE2+RTE3	MLP	54.6%
RTE1	SVM	54%
RTE3-With NER Module	SVM	53.8%
Baseline	-	50%

Table 2.Results obtained in three-way classification task using Stanford datasets.

Here we noted that using RTE3 instead of RTE2 or RTE1 in both classification tasks (two and three way) 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) shown that RTE3 alone could produce higher results that training on RTE3 merged with RTE2 for the two-way task. Consequently, 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 worse results, maybe because this dataset has been collected with different text processing applications (QA, IE, IR, SUM, PP and MT), and our system do not have into account it.

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).

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 of 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 neither classification task, in average SVM is one of the best classifiers.

We have to remark that in two-way task we obtained a difference of 3.8% between the best and worst combination of datasets and classifiers; meanwhile, 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 data set and classifier has more impact over 2-way task than over 3-way task.

The performance in all the cases was clearly above those baselines. Only using BPI in two-way classification we obtained a worse result than baseline, and it is because BPI is syntactically simpler than PASCAL RTE; therefore, it seems that is not good enough 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 the misclassified instances provide evidence that could be improved almost up to 80% (e.g: improving the acronym database), and having into account the coverage of corpus that was 10%, it could impact positively on the overall performance of the system. While this Name Entity Preprocessing module approach performed reasonably well in these evaluations, we feel that even better results could be obtained by adding heuristic rules and knowledge base information.

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:

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	-

Table 3.Correlation matrix of features.

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 by eliminating feature (2).

Additionally, we calculated the Kappa statistics over all development set using WEKA (Witten and Frank, 2000) for both 2-way and 3-way task classification. The average for Kappa measure was 0.138 for two-way task and 0.168 for three-way task.

In general, because of the corpus was incremented we obtained better values for Kappa. Nevertheless, the best value was obtained with RTE-3 two ways using MLP. In this case, the Kappa measure was 0.35 for cross validation experiment (See Tables 4 and 5).

There are two main reasons because of the values were slight: the size of the corpus and the mistakes made in the class contradiction, which was the most difficult class to predict in the 3-way classification.

Finally, we assessed 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.

Dataset	**Classifier**	**Accuracy%**
RTE3	MLP	**65.5%**
RTE2 + RTE3	MLP	60.68%
RTE1+RTE2+RTE3	MLP	59.35%
RTE2	SVM	56.62%
RTE1+RTE2	SVM	55.84%
RTE1	Decision tree	54.70%

Table 3.Results obtained with Cross Validation in three-way task.

Dataset	Classifier	Accuracy%
RTE3	BayesNet	**67.85%**
BPI	BayesNet	64%
RTE1 + RTE2 + RTE3	MLP	63.16%
RTE2	SVM	60.12%
RTE1+RTE2	MLP	59.79%
RTE1	SVM	57.83%

Table 4.Results obtained with Cross Validation in two-way task.

The results on test set are worse than those obtained on training set, which is most probably due to the overfitting of classifiers and because of the possible difference between these datasets.

4. Conclusion and Future Work

We presented our RTE system that is based on a wide range of machine learning classifiers. It was a workbench that gave us a vision and knowledge about the structure of the data set and the abilities of different classifiers to learn them.

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 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 suggests that further refinements over heuristic rules can achieve better results.

Despite not presenting an exhaustive comparison among all available datasets and classifiers, we can conclude that the best combination of RTE-s datasets and classifiers chosen for two way task produce more impact that the same combination for three way task, almost for all experiments that we did. In fact, the use of RTE3 alone improved the performance of our system.

Finally, 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 test the improvements they may yield. Additional work will focused on improving the performance of our NE preprocessing module.

5. References

[1] Prodromos Malakasiotis and Ion Androutsopoulos. *Learning Textual Entailment using SVMs and String Similarity Measures.* ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, (ACL 2007), Prague, Czech Republic, 2007.

[2] Julio Javier Castillo, and Laura Alonso i Alemany. *An approach using Named Entities for Recognizing Textual Entailment.* TAC 2008, Gaithersburg, Maryland, USA, November 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: Computer Science and Computational Biology.* 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 (2005). *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[7] Alvaro Rodrigo, Anselmo Peñas, Jesus Herrera, Felisa Verdejo. *Experiments of UNED at the Third RTE* Challenge. Proceedings of the ACL-PASCAL 2007.

[8] British Atmospheric Data Centre (BADC) acronym database: *http://badc.nerc.ac.uk/help/abbrevs.html*

[9] D. Inkpen, D. Kipp and V. Nastase. *Machine Learning Experiments for Textual Entailment.* Proceedings of the second RTE Challenge, Venice-Italy, 2006.

[10] Bill Dolan, Chris Quirk, and Chris Brockett. 2004. *Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources.* In COLING '04: Proceedings of the 20th international conference on Computational Linguistics, page 350, Morristown, NJ, USA. Association for Computational Linguistics.

[11] F. Zanzotto, Marco Pennacchiotti and Alessandro Moschitti. *Shallow Semantics in Fast Textual Entailment Rule Learners*. In Proceedings of the Third Recognizing Textual Entailment Challenge, Prague, 2007.

[12] Marie-Catherine de Marneffe, et al. Manning.*Learning to distinguish valid textual entailments.* In Proceedings of the Third Recognizing Textual Entailment Challenge, Italy, 2006.