An approach using Named Entities for Recognizing Textual Entailment

Julio Javier Castillo, Laura Alonso i Alemany
Faculty of Mathematics Astronomy and Physics
Department of Computer Science
National University of Cordoba, Argentina
{cj, alonso}@famaf.unc.edu.ar

Abstract. This paper describes the Sagan system in the context of the Fourth Pascal Recognizing Textual Entailment (RTE-4) Evaluation Challenge.
Sagan applies a Support Vector Machine classifier to examples characterized by four features based on: edit distance, distance in WordNet and Longest Common Substring between text and hypothesis. Additionally, we created a filter applying hand-crafted rules based on Named Entities to detect cases where no entailment was found.
Despite this simple approach, results are promising. Applying the Named Entity filter yields a small improvement in precision.

1. Introduction

The objective of the Recognizing Textual Entailment Challenge task is to automatically determine whether or not a hypothesis(H) can be inferred from a text (T).

This year, RTE-4 has posed two different tracks: one involving the traditional two-way distinction between entailment and non-entailment, and another where a three-way distinction was intended, between *entailment*, *contradiction* and *unknown*, when no information to accept or reject the hypothesis is provided in the text.

The Sagan[1] system applies a Support Vector Machine approach to the problem of recognizing textual entailment. Only four simple features are used to characterize the relationship between text and hypothesis for both training and test cases:

- Levenshtein distance between T and H (over stems).
- Lexical similarity (based on Levenshtein distance).
- Semantic distance as calculated via WordNet.
- Longest common substring.

Additionally, we have developed a filter that detects cases where no entailment relation is found. This filter applies simple, hand-crafted rules about Named Entities found in the text and hypothesis. It has coverage of about 10% of the cases, and yields a small improvement in the performance of the system.

Two runs were submitted to TAC'08: one with the system including the Named Entity filter, and another without it, to assess the impact of the filter in the overall performance of the system.

The rest of the paper is as follows. We describe the architecture of our system in Section 2.The results of experimental evaluation are discussed in Section 3. Finally, we present some conclusions and lines for future work to improve our system's performance.

2. System description

The Sagan system is based on a SVM approach to recognizing textual entailment for English. Two runs were submitted to the TAC 2008 Challenge, differing only on the treatment of Named Entities. Figure 1 shows the general architecture of our system.

[1] In honour to Carl Sagan.

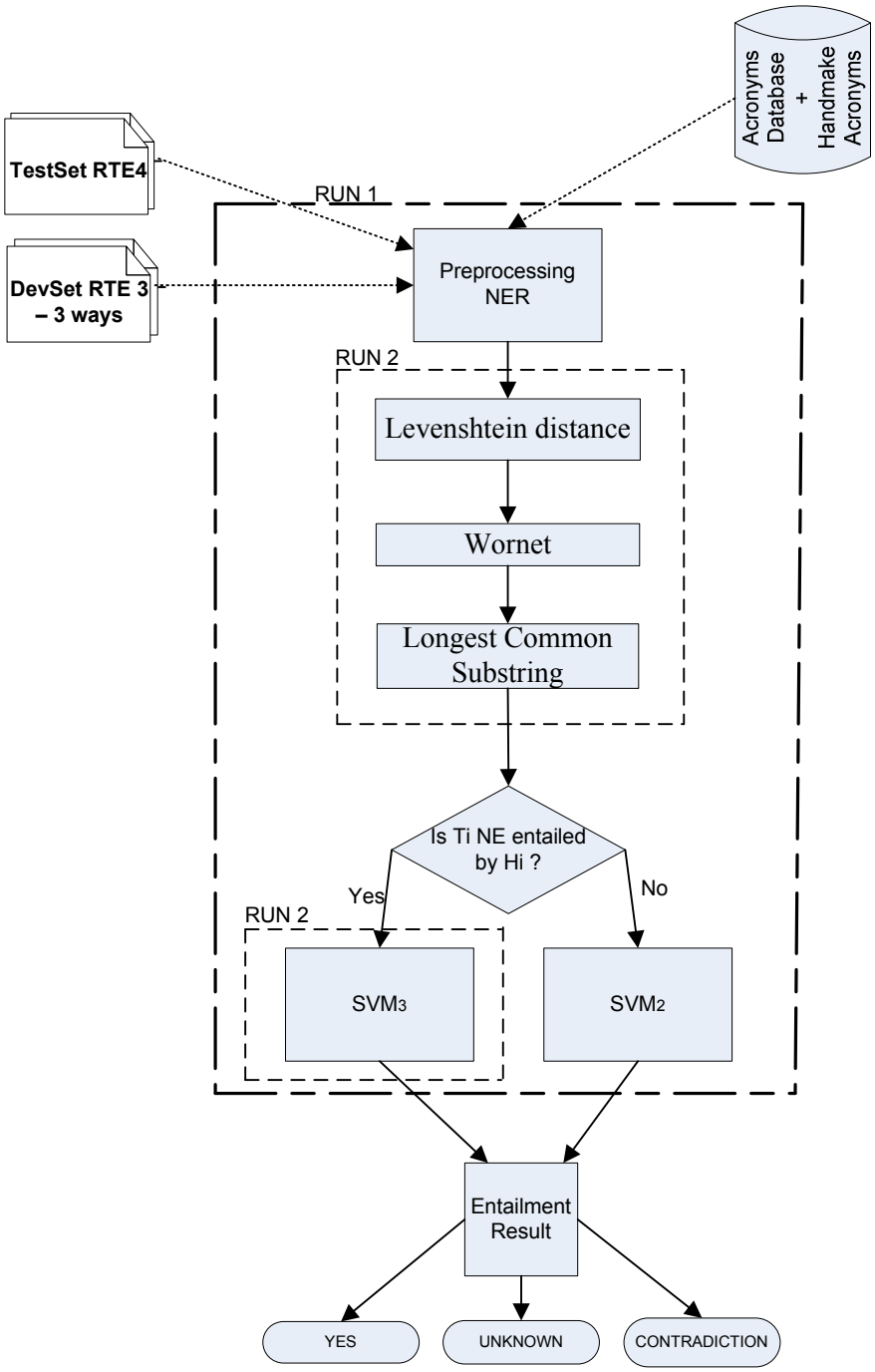

Figure 1. The Sagan System for Recognizing Textual Entailment for English.

Our system is an extension of our preliminary system that we presented [7] in the Answer Validation Exercise (AVE) 2008, part of CLEF'08.

Text-hypothesis pairs are characterized with four main features: the Levenshtein distance between each other, lexical distance based on Levenshtein, a semantic distance based on WordNet and their Longest Common Substring. Then, a Support Vector Machine is used to classify test pairs in three classes: *entailment*, *contradiction* or *unknown*. The SVM was trained with the tree-way corpus piloted in the RTE3 Pascal Challenge[2].

[2] http://nlp.stanford.edu/RTE3-pilot/

Text-hypothesis pairs are stemmed with Porter's stemmer[3] [3] and PoS tagged with the tagger in the OpenNLP[4] framework.

Additionally, a NER-based filter was added to the system. This performs NER in text-hypothesis pairs and applies a series of hand-crafted rules to discard that an entailment relation is actually found in the pair. If that is the case, a specialized SVM_2 is applied to classify the pair between contradiction and unknown. The two runs submitted to TAC for evaluation differed on their use for this filter: RUN1 did use the filter, RUN2 did not use it.

2.1 NER filter

Sagan 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. These pairs are classified by a specialized SVM_2 as *contradiction* or *unknown*.

This module applies hand-crafted rules to the output of the automated NER analysis within OpenNLP. NER recognition was enhanced by using an acronym database[5].

The form of the rules is as follows: for each type of NE (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 will therefore should be classified as either *contradiction* or *unknown*.

In a study on the RTE3 and RTE4 training corpus, these rules applied to approximately 10 percent of the text-hypothesis pairs. The accuracy of the filter, as evaluated in TAC'08, is 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-RTE 4):

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 hand-crafted rules.

2.2. Lexical Distance

We use the standard Levenshtein distance [10] 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 to introduce 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 as follows:

- 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 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.
The maximum value of TotalSim is equal to *Max (Length (Text), Length (H))*.
Length (T) is the number of tokens in T.
Length (H) is the number of tokens in H.

[3] http://tartarus.org/~martin/PorterStemmer/
[4] http://opennlp.sourceforge.net/
[5] Acronym database : http://badc.nerc.ac.uk/help/abbrevs.html

2.3. WordNet Distance

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

1. Tokenization
2. Stemming
3. PoS tagging
4. Word sense disambiguation using the Lesk algorithm [8], based on Wordnet definitions.
5. 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.
6. To obtain the final score, we use matching average.

The semantic similarity between two words (step 5) 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 subsumer of *s* and *t*.

The matching average (step 6) 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 [9] will find the longest strings which are substrings of both T and H. It is founded on dynamic programming.

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

In all practical cases, $\min(Length(T), Length(H))$ will be equal to $Length(H)$.

Before performing LCS, texts were tokenized and stemmed.

3. Experimental Evaluation

Two runs were submitted to TAC'08 for evaluation, one using the Named Entity filter (Run 1) and the other not using it (Run 2).

Two baselines are provided to better assess the performance of the system in both runs: a uniform baseline that randomly assigns *entailment*, *contradiction* or *unknown* to test cases and a weighted baseline that randomly assigns these three categories but following the probabilities that are found in the gold standard test set RTE 4, that is: 0.5 for *entailment*, 0.35 for *unknown* and 0.15 for *contradiction*.

For 2-way classification, the uniform random baseline was 0.5, Run1 obtained 0.576 and Run2 obtained 0.571, that is, the impact of the Named Entity filter introduced a very slight increase in the performance of the system. The performance of both runs was close to the baseline, although clearly above it.

For 3-way classification, the accuracy of the uniform random baseline was 0.33, while the accuracy of the weighted baseline was 0.39. Both Run1 and Run2 performed significantly better than the baselines: 0.538 and 0.546, respectively. In this case, the Named Entity filter introduced a slight decrease in performance.

Table 1 shows the results of both runs, and table 2 shows the results separated by run and task.

	Acc – 2 way	Acc – 3 way
RUN 2	0.571	**0.546**
RUN 1	**0.576**	0.538
Weighted Random	0.50	0.390
Uniform Random	0.50	0.333

Table 1: General 2 and 3ways classification evaluation of the Sagan system.

	Accuracy			
Task	RUN 1 – 3 ways – With Preprocessing NER module	RUN 1 –2 ways - With Preprocessing NER module	RUN 2 – 3 ways – Without NER module	RUN 2 – 2 ways - Without NER module
IR	0,6333	**0,6900**	0,6467	0,6833
QA	0,4700	**0,4850**	0,4650	0,4750
SUM	0,5950	**0,6250**	0,6000	0,6200
IE	0,4500	**0,4900**	0,4633	**0,4900**

Table 2: Results of Sagan system separated by task and run.

4. Conclusion and Future Work

We presented at TAC 2008 our RTE system that is based on a SVM classifier. We have used Levenshtein distance, lexical similarity, semantic similarity with Wordnet and Longest Common Substring. Additionally, we have assessed a small impact of a Named Entity rule-based filter in the performance of the system.

In spite of the simplicity of the approach, we have obtained a reasonable 0.576 of accuracy for first run in 2-way task, and 0.546 of accuracy for second run in 3-way task.

Future work is oriented to analyze and enhance our Named Entity filter. We will also experiment with different classifiers, such as Bayesian Binary Regression (BBR), and will use different training sets, possibly from previous RTE and AVE. To enhance the system, we will work with lexical and semantic similarity, adding features and testing the improvements they may yield.

References

[1] Alvaro Rodrigo, Anselmo Peñas, Jesus Herrera, Felisa Verdejo. *Experiments of UNED at the Third Recognizing Textual Entailment Challenge*. Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. 2007.

[2] Prodromos Malakasiotis and Ion Androutsopoulos. *Learning Textual Entailment using SVMs and String Similarity Measures*. ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007), Prague, Czech Republic, 2007.

[3] Julie Beth Lovins. *Development of a stemming algorithm*. Mechanical Translation and Computational Linguistics, March 1968.

[4] Corinna Cortes and V. Vapnik, *Support-Vector Networks*, Machine Learning, 20, 1995.

[5] Peñas A., Rodrigo A., Sama V., and Verdejo F. *Overview of the Answer Validation Exercise 2006*, In Workingnotes for the Cross Language Evaluation Forum Workshop (CLEF 2006), Alicante, España, September 2006.

[6] Peñas A., Rodrigo A., Sama V., and Verdejo F. *Overview of the Answer Validation Exercise 2007*, In Working notes for the Cross Language Evaluation Forum Workshop (CLEF 2007), Budapest, Hungary, September 2007.

[7] Castillo, Julio Javier. *The Contribution of FaMAF at QA@CLEF 2008*. Answer Validation Exercise. In Working notes for the Cross Language Evaluation Forum Workshop (CLEF 2008), September, Aarhus, Denmark, September 2008.

[8] M. Lesk. *Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone*. In Proceedings of SIGDOC '86, 1986.

[9] Gusfield, Dan. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, year 1999.

[10] V. Levenshtein. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals*. Soviet Physics Doklady, 10:707, February 1966.