Sagan in TAC2009: Using Support Vector Machines in Recognizing Textual Entailment and TE Search Pilot task

Julio Javier Castillo

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

Abstract

This paper describes the Sagan system in the context of the Fifth Pascal Recognizing Textual Entailment (RTE-5) Evaluation Challenge and the new Textual Entailment Search Pilot Task.

The system employs a Support Vector Machine classifier with a set of 32 features, which includes lexical and semantic similarity for both two-way and three-way classification tasks.

Additionally, we show an approach to deal with the problem of search entailment in a context of a set of document that uses co-reference analysis.

Keywords

Textual entailment, machine learning, lexical features.

1. Introduction

The goal of the RTE Track is to develop systems that recognize when one piece of text(T) entails another(H).

This year the National Institute of Standards and Technology (NIST) organized the Text Analysis Conference (TAC) 2009, which has three main tracks, namely Knowledge Base Population (KBP), Recognizing Textual Entailment (RTE), and Summarization, providing a common evaluation framework of different NLP tasks.

In order to move the RTE task towards more realistic application scenarios the texts will come from a variety of sources and may include typographical errors and ungrammatical sentences. This time, RTE5 will be based on only three application settings: QA, IE, and IR.

In addition to the main task is offered a new Textual Entailment Search pilot that is situated in the summarization application setting, where the task has the goal of finding all Texts in a set of documents that entail a given Hypothesis.

In this paper we present the Sagan system as part of Famaf participation in the textual entailment recognition main task and textual entailment search pilot task.

The Sagan system applies a Support Vector Machine approach to the problem of recognizing textual entailment.

This year, we modify our past Sagan system in order to work almost exclusively with lexical features, with the aims of exploring more deeply how lexical information could help in the RTE task. Then, we use 31 lexical features and only 1 semantic feature based on WordNet.

These features are used to characterize the relationship between text and hypothesis for both training and test cases.

The remainder of the paper is organized as follows: Section 2 describes the architecture of our system, whereas Section 3 shows the experimental evaluation and discussion of them.

Finally, Section 4 summarizes the conclusions and lines for future work.

2. System Architecture

This section provides an overview of our system as used for RTE5 track at the TAC 2009 Challenge. The system is based on a machine learning approach for recognizing textual entailment.

We use a supervised machine learning approach to train a SVM classifier over a variety of lexical and semantic metrics. Every output of the metrics is treat as a feature and used in the training step taking the previous RTE's datasets.

In Figure 1 we present a brief overview of the Sagan system.

First, the <T,H> pairs are pre-processed with optional modules, as described next.
Second, we compute 32 features which belong to two different categories lexical and semantics metrics.
Finally, for every submitted runs we use a SVM classifier for 2-way and 3-way classification tasks
Using a machine learning approach we tested with SVM classifier in order to classify RTE-5 test pairs in three classes: entailment, contradiction or unknown.

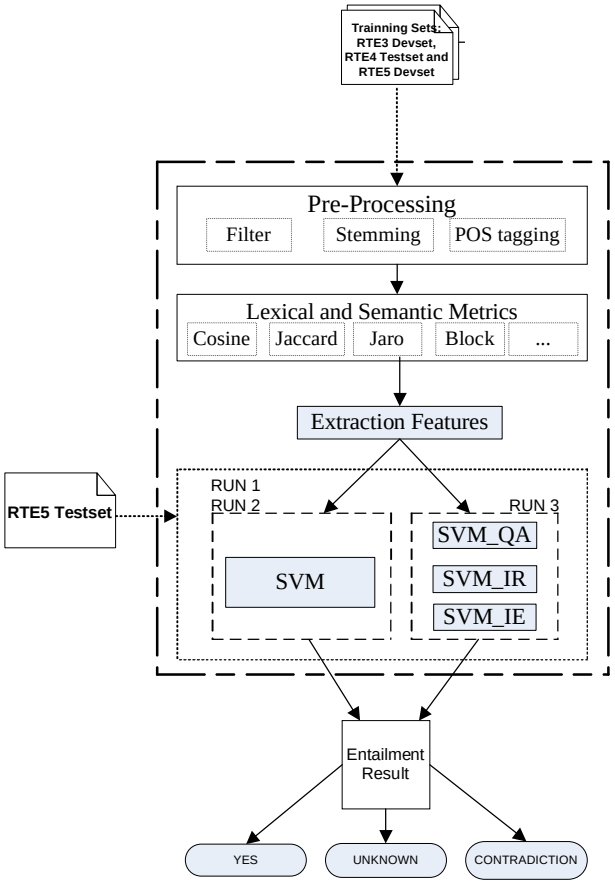

Figure 1.General architecture of our Sagan system for RTE5.

2.1 Preprocessing

The Preprocessing module has three optional submodules as needed by the different features:

Tokenizer: The text-hypothesis pairs are tokenized with the tokenizer of OpenNLP framework.

Stemmer: Text-hypothesis pairs are stemmed with Porter's stemmer[1] [3].

Tagger: Text-hypothesis pairs are PoS tagged with the tagger in the OpenNLP[2] framework.

Three runs were submitted to the Textual Analysis Conference 2009 differing only on the preprocessing stage.

For RUN1 we use 800 pairs of the RTE3 devset, 1000 pairs of the RTE4 testset, and 600 pairs of the RTE5 devset as training set. Therefore 2400 pairs are used for train purpose.

The RUN1 is trained with the union of the following datasets: RTE3 devset + RTE4 testset + RTE5 devset.

On the other hand, RUN2 is trained with the union of the datasets: RTE3 devset + RTE4 testset + RTE5 devset, but without SUM sample test pairs. Therefore here 2000 pairs are used as training set.

Finally, RUN3 is the result of apply three Support Vector Machines: SVM_QA, SVM_IR, and SVM_IE, trained over RTE3-DS + RTE4-TS +RTE5-DS.

The SVM_QA is a SVM that is trained only using the pairs of QA task over the datasets: RTE3 devset + RTE4 testset + RTE5 devset.

In the same way, SVM_IR and SVM_IE are trained only using IR and IE pairs, respectively.

The training set for RUN3 is composed by 600 QA-pairs, 700 IE-pairs, and 700 IE-pairs. Table 1 shows the training set composition used for every SVM.

Datasets	*Pairs*	*Total Pairs*
RTE3-DS_QA	200	
RTE4-TS_QA	200	
RTE5-DS_QA	200	
Total QA pairs		*600*
RTE3-DS_IE	200	
RTE4-TS_IE	300	
RTE5-DS_IE	200	
Total IE pairs		*700*
RTE3-DS_IR	200	
RTE4-TS_IR	300	
RTE5-DS_IR	200	
Total IR pairs		*700*

Table 1.Training set composition for QA, IR and IE – SVM's.

The motivation of the input features was to test our system over a wide range of lexical feature and try to determinate whether this approach could improve our performance.

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

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

2.2 Features

We use a supervised machine learning approach to train a classifier over a variety of lexical and semantic metrics. Thus, we use the output of each metric as a feature, and train a SVM classifier.

For this purpose, we use 32 features/metrics over Text (T) and Hypothesis (H) as explained below.

The first 12 features do not require additional explanation.

(1) Percentage of Words of Hypothesis in the text.

(2) Percentage of word of text in hypothesis.

(3) Percentage of bigrams of Hypothesis in Text.

(4) Percentage of trigrams of hypothesis in the text.

(5) TF-IDF Measure.

(6) Standard levenshtein distance [5] (character based).

(7) Percentage of words of Hypothesis in the text.

(8) Percentage of words of text in Hypothesis (over stems).

(9) Percentage of bigrams of hypothesis in the Text (over stems)

(10) Percentage of trigrams of Hypothesis in Text (over stems).

(11) TF-IDF measure (over stems).

(12) Levenshtein distance (over stems).

(13) String similarity using Levenshtein distance using Wordnet as defined in (Castillo Julio J., et al. 2008).

(14) Semantic similarity using WordNet (Castillo Julio J., et al. 2008).

(15) 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. Before performing LCS, texts were tokenized and stemmed.

(16) Block distance.

(17) Chapman length deviation.

(18) Chapman mean length.

(19) Cosine similarity.

(20) Dice similarity.

(21) Euclidean distance.

(22) Jaccard similarity.

(23) Jaro.

(24) Jaro Winkler.

(25) Matching coefficient.

(26) Monge Elkan distance [11]:

(27) Needleman Wunch distance.

(28) Overlap Coefficient.

(29) QGrams distance.

(30) Smith-Waterman distance.

(31) Smith-Waterman with gotoh.

(32) Smith-Waterman with gotoh windowed affine.

The features 1 to 5, 7 and 16 to 32 were treated as bags of words, on the other hand, features 8 to 12 were treated as bags of stems.

The features 16 to 32 were calculated using SimMetrics[3] Library over string T and H, and following the traditional definition for every one of them.

2.3 Textual Entailment Search Pilot Task

In order to move towards more realistic scenarios and start to test RTE systems against real data, textual entailment search is proposed.
Thus, Textual Entailment Search Pilot task has the goal of analyzing the potential impact of textual entailment recognition on a real NLP application task.
The Textual Entailment Search task consists in finding all the sentences in a set of documents that entail a given Hypothesis.
The systems must find all the entailing sentences (Ts) in a corpus of 10 newswire documents about a common topic.
So, the main difference with respect to the main task is that in the Entailment Search task both Text and Hypothesis are to be interpreted in the context of the corpus.

[3] http://sourceforge.net/projects/simmetrics/

In this proposal, we propose a textual entailment search task based on coreference analysis. The assumption is that using coference analysis we will be able to recognize true and false entailment in the context of the corpus in which T and H belongs. As coreference tool we use OpenNlp toolkit.

The system Sagan has an extension to deal with Textual Entailment Search problem. It is a new module that performs the following algorithm:

1) Append a Hypothesis h_i to the document D_j.
2) Computes a coreference analysis over all document D_j.
3) Identify all coreferences that refer to the same entity.
4) Take the longest reference and replace all occurrences in the document.
5) Repeat for every Topic, Document and Text.

Example: The following example is extracted from the RTE Search Pilot Devset.

[French President Jacques Chirac, 16]
[Chirac, 16]

Where the first string represents the noun phrase that is being referenced and second number is a reference id.

Thus, the algorithm selects "French President Jacques Chirac" and replaces all references with the same id, using this noun phrase.
Sometimes, the result won't be a correct syntactically sentence. However, it will be human understandable. We expect that the overall sense of the sentence won't be changed.

Once, this process is performed every <T,H> pair of a document is taken and feed into the Sagan system such as explained before, following the RUN1 preprocessing procedure but with outputs True/False.

3. Experimental Evaluation and Discussion of Results

3.1 Results: RTE5 main task

Our official results for RTE5 testset for two-way and three-way classification task are summarized in Table 1.

Three runs were submitted to Textual Analysis Conference 2009 for evaluation and are shown on the table 1. Also, the high score and low score of the RTE5 participants and ablation test are shown below.

	Acc – 2 way	***Acc – 3 way***
Best System Score	0.7350	0.6833
Median Score 2-way	0.6117	---
Sagan1_abl-1	0.5517	0.53
RUN1	**0.5517**	**0.5217**
RUN2	0.545	0.52
Median Score 3-way	---	0.52
RUN3	0.5483	0.5183
Low Score	0.50	0.4383

Table 1.Results obtained with two-way and three-way classification task for RTE5 testset.

We note that training set for RUN1 consist of 2400 pairs, for RUN2 consist of 2000 pairs, and for RUN3 consist of 600 QA-pairs, 700 IR-pairs and 700 IE-pairs.

It suggest that RUN1 reach our best performance because of RUN1 has more samples <T,H> as training set, despite of the fact that includes SUM samples pairs.

However, RUN2 and RUN3 do not have a significant different with respect to RUN1.

For both, two-way and three-way task a slight and not statistical significant difference of 0.34% and 0.67% between the best and worst RUN is found, respectively.

The performance in all runs was clearly above those low scores; however our results are far of the best system score.

The RUN1 is trained using full RTE3 devset + RTE4 testset + RTE5 devset.

The best performance of our system was achieved with RUN1, and it was 55.17% and 52.17% of accuracy, for two and three way, respectively.

The accuracy of this run for two-way task is placed 5% below of median score. On the other hand, is placed 2.17% over the median score for three way task.

Thus, we conclude that this lexical approach is very preliminary and need to be improved of several ways.

An ablation test is a procedure that consists of "disconnect" one module (using a knowledge resource) of the system, in order to asses the contribution of that module to the overall accuracy of the system.

This year, ablation tests are mandatory for systems participating in the main task of RTE-5, with the aims of collecting data to better understand the impact of the

knowledge resources used by RTE systems and evaluate the contribution of each resource to systems' performance.

We perform an ablation test of "Wordnet" resource. It is implemented removing two features from the feature vector and working with 30 features. Wordnet resource has been ablated from RUN1.

First, features 13 and 14 were removed of the feature vector, and then rerunning the system on the test set. The results obtained are named as “Sagan1_abl-1” and shows in table1.

Interestingly, the ablation of these two features do not produces modification on two-way classification task and produces a very slight and not statistical significant increase on three-way task of 0.83%.

In addition, removing the feature 14 (the only one that deals with semantic similarity) does not impact on the overall classification.

In comparison with our last participation, we conclude that this semantic feature “lose relative-importance” having into account that positive impact of previous version of the Sagan system.

Table 2 shows the results obtained on RTE two-way and three-way classification task for every RUN and subtask.

Always the IR subtask yields the best results, maybe because this dataset is the easier subtask to predict.

Finally, we note that interestingly using four SVM one for each task we obtain similar results but using only 700 <T,H> pairs.

	Accuracy					
	RUN 1 – 3 ways	*RUN 1 – 2 ways*	*RUN 2 – 3 ways*	*RUN 2 – 2 ways*	*RUN 3 – 3 ways*	*RUN 3 – 2 ways*
IR	**0.655**	**0.695**	0.635	0.665	0.63	0.645
IE	0.41	0.44	0.41	0.435	**0.45**	**0.475**
QA	0.5	0.52	**0.515**	**0.535**	0.475	0.525

Table 2. Results of Sagan system divided by task and run.

3.2 Results: TE Search pilot task

Our official results for TE Search Pilot task are summarized in Table 1.

Together our submission, the high score and low score of RTE5 participants shown below.

	F-measure	***Precision***	***Recall***
High Score	0.4559	---	---
Median Score	0.3012	---	---
RUN1	**0.1816**	**0.1016**	**0.855**
Low Score	0.0955	---	---

Table 3. Result submission of Sagan system for Textual Entailment Search Pilot Task.

Eight teams submitted a total of 20 runs to this task. Our RUN is clearly above the system with low score, but is below average.

Despite of the fact that our very simple approach we think that a lot of improvements could be done in order to improve the F-score of the Sagan system, refining the before algorithm.

4. Conclusion and Future Work

In this paper we use a set of lexical features and try to determine how lexical information helps in the textual entailment semantic task.

We show the Sagan RTE system that performs two-way and three-way textual entailment. The best results are reached on the three-way task

We present our submission for the Recognizing Textual Entailment main track, and also we describes our participation in the textual entailment search pilot task

As conclusion, we need more balanced feature set using not only lexical features, but also syntactic and semantic features, in order to improve the accuracy of the system.

Additionally, we need to compute correlations between all features in order to avoid "redundant information" at the moment of characterizing the RTE task.

On the other hand, our approach to Textual Entailment Search is very simple and preliminary and need to be improved using knowledge resources and more in depth coreference analysis.

Future work is oriented to experiment with additional lexical, syntactic and semantic similarities features and test the improvements they may yield.

5. References

[1] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, Bill Dolan. The Third PASCAL Recognizing Textual Entailment Challenge. in Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 1–9, Prague, June 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] D. Inkpen, D. Kipp and V. Nastase. *Machine Learning Experiments for Textual Entailment.* Proceedings of the second RTE Challenge, Venice-Italy, 2006.

[7] 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.

[8] 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.

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

[10] Castillo, Julio. A Study of Machine Learning Algorithms for Recognizing Textual Entailment.RANLP2009, Borovets, Bulgaria, 2009.

[11] Eugene Agichtein et at. Combining Lexical, Syntactic, and Semantic Evidence for Textual Entailment Classification. TAC 2008, Gaithersburg, Maryland, USA, November 2008.