Procesamiento del Lenguaje Natural, Revista nº 44, marzo de 2010, pp 139-145 recibido 21-01-10 revisado 17-02-10 aceptado 05-03-10

An approach to Recognizing Textual Entailment and TE Search Task using SVM

Una aproximación al RTE y a la Tarea de Búsqueda de Implicación Textual usando Máquinas de Soporte Vectorial

Julio Javier Castillo
Faculty of Mathematic Astronomy and Physics
National University of Cordoba, Argentina
cj@hal.famaf.unc.edu.ar

Abstract: This paper shows a Recognizing Textual Entailment System, and a sub-system that address the Textual Entailment Search Task. This 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 dealing with the problem of searching entailment in a context of a set of documents that use co-reference analysis.
Keywords: Textual entailment, machine learning, lexical features.

Resumen: En este trabajo se presenta un sistema de Reconocimiento de Implicación Textual, y un subsistema que permite atacar el problema de la búsqueda de implicaciones. El sistema utiliza un clasificador de Máquina de Soporte Vectorial con un conjunto de 32 características, las cuales incluyen similaridad léxica y semántica para las tareas de clasificación de dos y tres vías. Adicionalmente, se presenta una aproximación inicial al problema de búsqueda de implicancias en un conjunto de documentos que utiliza análisis de correferencias.
Palabras clave: Implicancia Textual, aprendizaje automático, características léxicas.

1 Introduction

The objective of the Recognizing Textual Entailment Challenge is the task of determining whether the meaning of the Hypothesis (H) can be inferred from a text (T).

In RTE5 the texts comes from a variety of sources and includes typographical errors and ungrammatical sentences.

The RTE5 is based on only three application settings: QA, IE, and IR, in contrast to previous RTEs. There is a new Textual Entailment Search Pilot Task that is situated in the summarization application setting, where the task has the goal of finding all Texts in a set of documents that entails a given Hypothesis.

In this paper, we present a system that addresses the textual entailment recognition main task and textual entailment search pilot task. The system applies a Support Vector Machine (SVM) approach to the problem of recognizing textual entailment. Our system, 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.

ISSN 1135-5948

2 System Architecture

This section provides an overview of our system, which is based on a machine learning approach for RTE.

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 treated as a feature and used in the training step, taking the RTE3 devset, RTE4 annotated set, and RTE5 devset as training datasets. In Figure 1 we present a brief overview of the system.

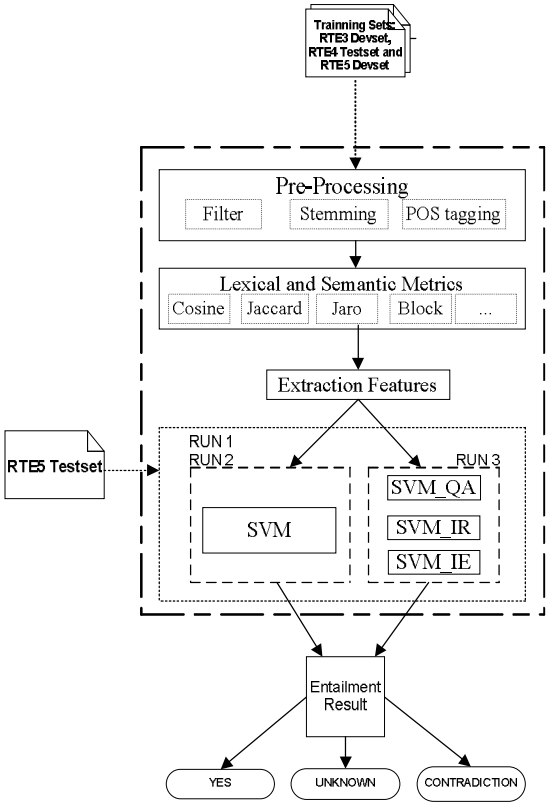

Figure 1: General architecture of the system for RTE5.

First, the <T,H> pairs are pre-processed with optional modules, as it is described later. Second, we compute 32 features that belong to two different categories: lexical and semantic metrics. Finally, 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.

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 stemmer1 (Lesk, 1986).

Tagger: Text-hypothesis pairs are PoS tagged with the tagger in the OpenNLP2 framework.

We tested three runs differing only in 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 training purposes. 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 these datasets: RTE3 devset + RTE4 testset + RTE5 devset, but without SUM sample test pairs. Therefore, 2000 pairs are used as training set.

Finally, RUN3 is the result of applying 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 by 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 by 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	

[1] http://tartarus.org/~martin/PorterStemmer/
[2] http://opennlp.sourceforge.net/

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 testing our system over a wide range of lexical features and trying to determinate whether this approach could improve our performance.

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 (Levenshtein, 1966) (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 based on Levenshtein distance using Wordnet as defined in (Castillo, 2008).
(14) Semantic similarity using WordNet (Castillo 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 (Levenshtein, 1966) 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 (Agichtein, et al., 2008).
(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 testing RTE systems against real data, textual entailment search is proposed. So, 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 of finding all the sentences in a set of documents that entails 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

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

with respect to the main task is that in the Entailment Search task both Text and Hypothesis have to be interpreted in the context of the corpus.

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

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

1) Appends a Hypothesis hi to the document Dj.
2) Computes a coreference analysis over all documents Dj.
3) Identifies all coreferences that refer to the same entity.
4) Takes the longest reference and replaces all occurrences in the document.
5) Repeats for every Topic, Document and Text.

The following example shown as an entity will be replaced by an equivalent entity adding redundant information. The example was 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 the second number is a unique reference id in the whole document.

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.

Also, it is important to note that the previous algorithm, in some cases, could transform the hypothesis. For example, the hypothesis 28 in the testset is transformed as:

H28: "Bobby Fischer faced deportation to the United States."

H28-modified: "Fischer, an outspoken critic of the U.S government, faced deportation to the United States."

So, from a semantic point of view, the H28 provides more information but is equivalent for our textual entailment task because we replace the same entity in all occurrences in the document.

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

2.3.1. Features used in Textual Entailment Search Task

In the Textual Entailment Search Task we use only four features numbered, which are (12), (13), and (14) and (15), as explained before.

We choose only a limited set of features, due to computational constrains; this is because processing all coreferences for all documents it is a very slow process.

The motivation of the input features:
Levenshtein distance (12) is motivated by the good results obtained as a measure of similarity between two strings. Using stems, this measure improves the Levenshtein over words. The lexical distance feature (13) based on Levenshtein distance is interesting because works to sentence level. Semantic similarity using WordNet (14) is interesting because of the capture of the semantic similarity between T and H to sentence level. Longest common substring (15) is selected because it is easy to implement and provides a good measure for word overlap.

3 Experimental Evaluation and Discussion of Results

3.1. Results: RTE5 main task

Our results for RTE5 testset for two-way and three-way classification task are summarized in Table 2.

In this table, we compare our results with those obtained by a selected set of systems that submitted their results to Textual Analysis Conference 2009, which is a common framework of evaluation. Thus, the high score and low score of the RTE5 participants and ablation test are shown below.

RTE Systems	Acc - 2way	Acc - 3 way
Best System Score	0.7350	0.6833
Median Score 2-way	0.6117	---
X1_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 2: Results obtained with two-way and three-way classification task for RTE5 testset.

We note that, RUN1 consists of 2400 pairs, RUN2 consists of 2000 pairs, and for RUN3 consists of 600 QA-pairs, 700 IR-pairs and 700 IE-pairs. This suggests that RUN1 reaches our best performance, because 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 difference 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. The performance, in all runs, was clearly above those low scores; however, our results are far from the best system score.

The RUN1 was trained using full RTE3 devset + RTE4 testset + RTE5 devset. Our best performance 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, it 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 on several ways.

An ablation test is a procedure that consists of "disconnecting" 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.

The ablation tests are very important because allows collecting data to better understanding of the impact of the knowledge resources used by RTE systems and evaluating the contribution of each resource to systems performance.

We performed an ablation test of "Wordnet" resource. It is implemented by 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 "X_abl-1" and shown in Table 2.

Interestingly, the ablation of these two features does not produce 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.

Table 3 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.

RTE Sys-tems	RUN 1-3w	RUN 1 - 2w	RUN 2 - 3w	RUN 2 - 2w	RUN 3 - 3w	RUN 3 - 2w
IR	0.65	0.69	0.63	0.67	0.63	0.65
IE	0.41	0.44	0.41	0.44	0.45	0.48
QA	0.5	0.52	0.52	0.54	0.48	0.53

Table 3: Accuracy results divided by task and run.

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

3.2. Results: Textual Entailment Search Task

We use Search Task Development Set (ST) and RTE3+ST as training set.

We take all true cases and automatically generate false cases based on the development set in order to build a balanced training set.

Table 4 summarizes our results for Textual Entailment Search Task. The high score and low score of RTE5 participants in TAC 2009 are shown below all together.

RTE Systems	F- Measure	Precision	Recall
High Score	0.4559	---	---
RUNs Median Score	0.3012	---	---
RUNnew	0,287	0,214	0,395
RUN1	0.1816	0.1016	0.855
Low Score	0.0955	---	---

Table 4: Result submission for Textual Entailment Search Pilot Task.

Also, we performed additional experiments testing 3 machine learning algorithms: Decision Trees, SVM, and Multilayer Perceptron. We show only the best results, which were obtained by using RTE3+ST as training set and SVM as classifier. This RUNnew is shown in the Table4.

In the TAC 2009, eight teams submitted a total of 20 runs to this task. The RUN1 is clearly above the system with low score, and the RUNnew is placed slightly in the Median Score of the RUNs.

Despite our very simple approach, we think that several improvements could be done in order to improve the F-score of the system, refining the before algorithm and using syntactic features and more semantic information.

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 our RTE system that performs two-way and three-way textual entailment. The best results are obtained on the three-way task. We present we address the Recognizing Textual Entailment main track, and also we describe an initial approach to the textual entailment search 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 by using knowledge resources and more in depth coreference analysis.

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

5 *References*

Giampiccolo, D. , Magnini B., Dagan I. and Dolan B. 2007. The Third PASCAL Recognizing Textual Entailment Challenge in Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 1–9, Prague.

Castillo, J. and Alonso L. 2008. An approach using Named Entities for Recognizing Textual Entailment. TAC 2008, Gaithersburg, Maryland, USA.

Lesk, M. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone. In SIGDOC.

Gusfield, Dan. 1999. Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. CUP.

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

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

Dolan, Bill, Quirk C. and Brockett C. 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, Association for Computational Linguistics. Morristown, NJ, USA.

Zanzotto, F., Pennacchiotti F. and Moschitti A. 2007. Shallow Semantics in Fast Textual Entailment Rule Learners. In Proceedings of the Third Recognizing Textual Entailment Challenge, Prague.

De Marneffe, M. et al. 2006. Manning.Learning to distinguish valid textual entailments. In Proceedings of the Third Recognizing Textual Entailment Challenge, Italy.

Castillo, J. 2009. A Study of Machine Learning Algorithms for Recognizing Textual Entailment.RANLP2009, Borovets, Bulgaria.

Agichtein, E. et al. 2008. Combining Lexical, Syntactic, and Semantic Evidence for Textual Entailment Classification. TAC 2008, Gaithersburg, Maryland, USA..