Machine Learning Based Semantic Inference: Experiments and Observations at RTE-3

Baoli Li[1], Joseph Irwin[1], Ernest V. Garcia[2], and Ashwin Ram[1]

[1] College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
baoli@gatech.edu
gtg519g@mail.gatech.edu
ashwin@cc.gatech.edu

[2] Department of Radiology
School of Medicine, Emory University
Atlanta, GA 30322, USA
Ernest.Garcia@emoryhealthcare.org

Abstract

Textual Entailment Recognition is a semantic inference task that is required in many natural language processing (NLP) applications. In this paper, we present our system for the third PASCAL recognizing textual entailment (RTE-3) challenge. The system is built on a machine learning framework with the following features derived by state-of-the-art NLP techniques: lexical semantic similarity (LSS), named entities (NE), dependent content word pairs (DEP), average distance (DIST), negation (NG), task (TK), and text length (LEN). On the RTE-3 test dataset, our system achieves the accuracy of 0.64 and 0.6488 for the two official submissions, respectively. Experimental results show that LSS and NE are the most effective features. Further analyses indicate that a baseline dummy system can achieve accuracy 0.545 on the RTE-3 test dataset, which makes RTE-3 relatively easier than RTE-2 and RTE-1. In addition, we demonstrate with examples that the current Average Precision measure and its evaluation process need to be changed.

1 Introduction

Textual entailment is a relation between two text snippets in which the meaning of one snippet, called the *hypothesis* (*H*), can be inferred from the other snippet, called the *text* (*T*). Textual entailment recognition is the task of deciding whether a given *T* entails a given *H*. An example pair (pair id 5) from the RTE-3 development dataset is as follows:

T: A bus collision with a truck in Uganda has resulted in at least 30 fatalities and has left a further 21 injured.
H: 30 die in a bus collision in Uganda.

Given such a pair, a recognizing textual entailment (RTE) system should output its judgement about whether or not an entailment relation holds between them. For the above example pair, *H* is entailed by *T*.

The PASCAL Recognizing Textual Entailment Challenge is an annual challenge on this task which has been held since 2005 (Dagan et al., 2006; Bar-Haim et al. 2006). As textual entailment recognition is thought to be a common underlying semantic inference task for many natural language processing applications, such as Information Extraction (IE), Information Retrieval (IR), Question Answering (QA), and Document Summarization (SUM), the PASCAL RTE Challenge has been gaining more and more attention in the NLP community. In the past challenges, various approaches to recognizing textual entailment have been proposed, from syntactic analysis to logical inference (Bar-Haim et al. 2006).

As a new participant, we have two goals by attending the RTE-3 Challenge: first, we would like to explore how state-of-the-art language techniques help to deal with this semantic inference problem; second, we try to obtain a more thorough knowledge of this research and its state-of-the-art.

Inspired by the success of machine learning techniques in RTE-2, we employ the same strategy in our RTE-3 system. Several lexical, syntactical, and semantical language analysis techniques are

Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 159–164,
Prague, June 2007.

explored to derive effective features for determining textual entailment relation. Then, a general machine learning algorithm is applied on the transformed data for training and prediction. Our two official submissions achieve accuracy 0.64 and 0.6488, respectively.

In the rest of this paper we describe the detail of our system and analyze the results. Section 2 gives the overview of our system, while Section 3 discusses the various features in-depth. We present our experiments and discussions in Section 4, and conclude in Section 5.

2 System Description

Figure 1 gives the architecture of our RTE-3 system, which finishes the process of both training and prediction in two stages. At the first stage, a *T-H* pair goes through language processing and feature extraction modules, and is finally converted to a set of feature-values. At the second stage, a machine learning algorithm is applied to obtain an inference/prediction model when training or output its decision when predicting.

In the language processing module, we try to analyze *T-H* pairs with the state-of-the-art NLP techniques, including lexical, syntactical, and semantical analyses. We first split text into sentences, and tag the Part of Speech (POS) of each word. The text with POS information is then fed into three separate modules: a named entities recognizer, a word sense disambiguation (WSD) module, and a dependency parser. These language analyzers output their own intermediate representations for the feature extraction module.

We produce seven features for each *T-H* pair: lexical semantic similarity (LSS), named entities (NE), dependent content word pairs (DEP), average distance (DIST), negation (NG), task (TK), and text length (LEN). The last two features are extracted from each pair itself, while others are based on the results of language analyzers.

The resources that we used in our RTE-3 system include:

OAK: a general English analysis tool (Sekine 2002). It is used for sentence splitting, POS tagging, and named entities recognition.

WordNet::SenseRelate::Allwords package: a word sense disambiguation (WSD) module for assigning each content word a sense from WordNet (Pedersen et al., 2005). It is used in WSD module.

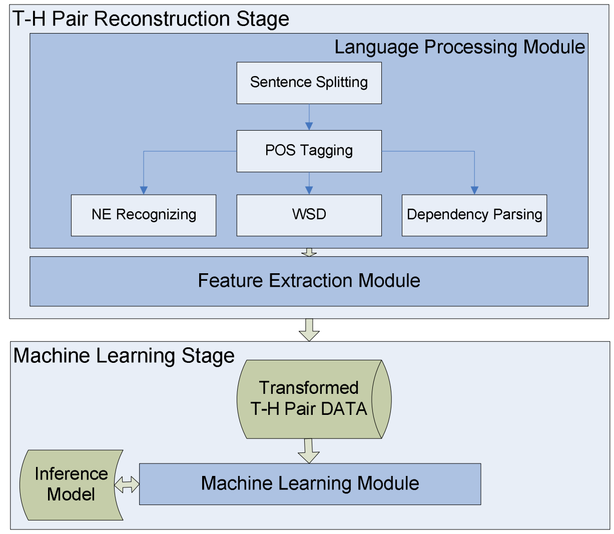

Figure 1. System Architecture.

WordNet::Similarity package: a Perl module that implements a variety of semantic similarity and relatedness measures based on WordNet (Pedersen et al., 2005). This package is used for deriving LSS and DIST features in feature extraction module.

C&C parser: a powerful CCG parser (Clark and Curran 2004). We use C&C parser to obtain dependent content word pairs in dependency parsing module.

WEKA: the widely used data mining software (Witten&Frank 2005). We have experimented with several machine learning algorithms implemented in WEKA at the second stage.

3 Features

In this section, we explain the seven features that we employ in our RTE-3 system.

3.1 Lexical Semantic Similarity (LSS)

Let $H=\{HW_1, HW_2, \ldots, HW_m\}$ be the set of words in a *hypothesis*, and $T=\{TW_1, TW_2, \ldots, TW_n\}$ the set of words in a *text*, then the lexical semantic similarity feature *LSS* for a *T-H* pair is calculated as the following equation:

$$LSS(H,T) = \frac{\sum_i \left(MAX_j \left(\frac{SSim(HW_i, TW_j)}{SSim(HW_i, HW_i)} \right) * IDF(HW_i) \right)}{\sum_i IDF(HW_i)}. \quad (1)$$

where $IDF(w)$ return the Inverse Document Frequency (IDF) value of word w, and $SSim$ is any function for calculating the semantic relatedness between two words. We use WordNet::Similarity package to calculate the semantic similarity of two content words in WordNet (Fellbaum 1998). This package provides many different semantic relatedness measures. In our system, we use the *Lesk* relatedness measure for function *SSim*, as it can be used to make comparisons between concepts of different parts of speech (POS) (Banerjee&Pedersen, 2002). Because the value of *SSim* may be larger than 1, we normalize the original value from the WordNet::Similarity package to guarantee it fall between 0 and 1.

For the words out of WordNet, e.g. new proper nouns, we use the following strategy: if two words match exactly, the similarity between them is 1; otherwise, the similarity is 0.

It needs to be pointed out that Equation (1) is a variant of the text semantic similarity proposed in (Mihalcea et al. 2006). However, in Equation (1), we take into account out of vocabulary words and normalization for some word-to-word similarity metrics that may be larger than 1.

In addition, we use an IDF dictionary from MEAD (Radev et al. 2001; http://www.summarization.com/mead/) for retrieving the IDF value for each word. For the words out of the IDF dictionary, we assign a default value 3.0.

3.2 Named Entities (NE)

Named Entities are important semantic information carriers, which convey more specific information than individual component words. Intuitively, we can assume that all named entities in a hypothesis would appear in a textual snippet which entails the hypothesis. Otherwise, it is very likely that the entailment relation in a *T-H* pair doesn't hold. Based on this assumption, we derive a NE feature for each *T-H* pair as follows:

$$NE(H,T) = \begin{cases} 1 & ,if\ |NE\_S(H)| = 0, \\ \dfrac{|NE\_S(H) \cap NE\_S(T)|}{|NE\_S(H)|} & ,if\ |NE\_S(H)| > 0. \end{cases}$$

Function NE_S derives the set of named entities from a textual snippet. When we search in *T* the counterpart of a named entity in *H*, we use a looser matching strategy: if a named entity *neA* in *H* is consumed by a named entity *neB* in *T*, *neA* and *neB* are thought to be matched. We use the English analysis tool OAK (Sekine 2002) to recognize named entities in textual snippets.

3.3 Dependent Content Word Pairs (DEP)

With the NE feature, we can capture some local dependency relations between words, but we may miss many dependency relations expressed in a long distance. These missed long distance dependency relations may be helpful for determining whether entailment holds between *H* and *T*. So, we design a DEP feature as follows:

$$DEP(H,T) = \begin{cases} 1 & ,if\ |DEP\_S(H)| = 0, \\ \dfrac{|DEP\_S(H) \cap DEP\_S(T)|}{|DEP\_S(H)|} & ,if\ |DEP\_S(H)| > 0. \end{cases}$$

Function DEP_S derives the set of dependent content word pairs from a textual snippet. We require that the two content words of each pair should be dependent directly or linked with at most one function word. We use C&C parser (Clark and Curran 2004) to parse the dependency structure of a textual snippet and then derive the dependent content word pairs. We don't consider the type of dependency relation between two linked words.

3.4 Average Distance (DIST)

The DIST feature measures the distance between unmapped tokens in the text. Adams (2006) uses a simple count of the number of unmapped tokens in the text that occur between two mapped tokens, scaled to the length of the hypothesis. Our system uses a different approach, i.e. measuring the average length of the gaps between mapped tokens. The number of tokens in the text between each consecutive pair of mapped tokens is summed up, and this sum is divided by the number of gaps (equivalent to the number of tokens – 1). In this formula, consecutive mapped tokens in the text count as gaps of 0, so a prevalence of consecutive mapped tokens lowers the value for this feature. The purpose of this approach is to reduce the effect of long appositives, which may not be mapped to the hypothesis but should not rule out entailment.

3.5 Negation (NG)

The Negation feature is very simple. We simply count the occurrences of negative words from a list in both the hypothesis (n_h) and the text (n_t). The list includes some common negating affixes. Then the value is:

$$NEG(H,T) = \begin{cases} 1, if\ n_h\ and\ n_t\ have\ the\ same\ parity \\ 0, otherwise \end{cases}$$

3.6 Task (TK)

The Task feature is simply the task domain from which the text-hypothesis pair was drawn. The values are Question Answering (QA), Information Retrieval (IR), Information Extraction (IE), and Multi-Document Summarization (SUM).

3.7 Text Length (LEN)

The Text Length feature is drawn directly from the length attribute of each *T-H* pair. Based on the length of *T*, its value is either "short" or "long".

4 Experiments and Discussions

We run several experiments using various datasets to train and test models, as well as different combinations of features. We also experiment with several different machine learning algorithms, including support vector machine, decision tree, k-nearest neighbor, naïve bayes, and so on. Decision tree algorithm achieves the best results in all experiments during development. Therefore, we choose to use decision tree algorithm (J48 in WEKA) at the machine learning stage.

4.1 RTE-3 Datasets

RTE-3 organizers provide two datasets, i.e. a development set and a test set, each consisting of 800 *T-H* pairs. In both sets pairs are annotated according to the task the example was drawn from and its length. The length annotation is introduced in this year's competition, and has a value of either "long" or "short." In addition, the development set is annotated as to whether each pair is in an entailment relation or not.

In order to aid our analysis, we compile some statistics on the datasets of RTE-3. Statistics on the development dataset are given in Table 1, while those on the test dataset appear in Table 2.

From these two tables, we found the distribution of different kinds of pairs is not balanced in both the RTE-3 development dataset and the RTE-3 test dataset. 412 entailed pairs appear in the development dataset, where 410 pairs in the test dataset are marked as "YES". Thus, the first baseline system that outputs all "YES" achieves accuracy 0.5125. If we consider task information (IE, IR, QA, and SUM) and assume the two datasets have the same "YES" and "NO" distribution for each task, we will derive the second baseline system, which can get accuracy 0.5450. Similarly, if we further consider length information (short and long) and assume the two datasets have the same "YES" and "NO" distribution for each task with length information, we will derive the third baseline system, which can also get accuracy 0.5450.

Long (135)	IE	NO	11	1.38%
		YES	17	2.13%
	IR	NO	22	2.75%
		YES	21	2.63%
	QA	NO	20	2.50%
		YES	27	3.38%
	SUM	NO	4	0.50%
		YES	13	1.63%
Short (665)	IE	NO	80	10.00%
		YES	92	11.50%
	IR	NO	89	11.13%
		YES	68	8.50%
	QA	NO	73	9.13%
		YES	80	10.00%
	SUM	NO	89	11.13%
		YES	94	11.75%

Table 1. Statistical Information of the RTE-3 Development Dataset.

Long (117)	IE	NO	11	1.38%
		YES	8	1.00%
	IR	NO	31	3.88%
		YES	23	2.88%
	QA	NO	13	1.63%
		YES	22	2.75%
	SUM	NO	4	0.50%
		YES	5	0.63%
Short (683)	IE	NO	84	10.50%
		YES	97	12.13%
	IR	NO	82	10.25%
		YES	64	8.00%
	QA	NO	81	10.13%
		YES	84	10.50%
	SUM	NO	84	10.50%
		YES	107	13.38%

Table 2. Statistical Information of the RTE-3 Test Dataset.

As different kinds of pairs are evenly distributed in RTE-1 and RTE-2 datasets, the baseline system for RTE-1 and RTE-2 that assumes all "YES" or all "NO" can only achieve accuracy 0.5. The relatively higher baseline performance for RTE-3 datasets (0.545 vs. 0.5) makes us expect that the average accuracy may be higher than those in previous RTE Challenges.

Another observation is that the numbers of long pairs in both datasets are very limited. Only

16.88% and 14.63% pairs are long in the development dataset and the test dataset respectively.

4.2 Evaluation Measures

Systems are evaluated by simple accuracy as in Equation (2); that is, the number of pairs (C) classified correctly over the total number of pairs (N). This score can be further broken down according to task.

$$Accuracy = \frac{C}{N} . \qquad (2)$$

There is another scoring available for ranked results, Average Precision, which aims to evaluate the ability of systems to rank all the *T-H* pairs in the test set according to their entailment confidence (in decreasing order from the most certain entailment to the least certain). It is calculated as in Equation (3).

$$AvgP = \frac{1}{R}\sum_{i=1}^{N}\frac{E(i) * Nep(i)}{i} . \qquad (3)$$

Where R is the total number of positive pairs in the test set, $E(i)$ is 1 if the i-th pair is positive and 0 otherwise, and $Nep(i)$ returns the number of positive pairs in the top i pairs.

RUN	Overall Accuracy	Accuracy by Task			
		IE	IR	QA	SUM
1	0.6400	0.5100	0.6600	0.7950	0.5950
2	0.6488	0.5300	0.6350	0.8050	0.6250

Table 3. Our Official RTE-3 Run Results.

4.3 Official RTE-3 Results

The official results for our system are shown in Table 3. For our first run, the model was trained on all the datasets from the two previous challenges as well as the RTE-3 development set, using only the LSS, NE, and TK features. This feature combination achieves the best performance on the RTE-3 development dataset in our experiments. For the second run, the model was trained only on the RTE-3 development dataset, but adding other two features LEN and DIST. We hope these two features may be helpful for differentiating pairs with different length.

RUN2 with five features achieves better results than RUN1. It performs better on IE, QA and SUM tasks than RUN1, but poorer on IR task. Both runs obtain the best performance on QA task, and perform very poor on IE task. For the IE task itself, a baseline system can get accuracy 0.525. RUN1 cannot beat this baseline system on IE task, while RUN2 only has a trivial advantage over it. In further analysis on the detailed results, we found that our system tends to label all IE pairs as entailed ones, because most of the IE pairs exhibit higher lexical overlapping between T and H. In our opinion, word order and long syntactic structures may be helpful for dealing with IE pairs. We will explore this idea and other methods to improve RTE systems on IE pairs in our future research.

Feature Set	Accuracy by Task				**Acc.**
	IE	IR	QA	SUM	
LSS	0.530	***0.660***	0.720	0.595	0.6263
NE	0.520	0.620	0.780	0.580	0.6250
DEP	0.495	0.625	0.575	0.570	0.5663
TK	0.525	0.565	0.530	0.560	0.5450
DIST	0.525	*0.435*	*0.530*	0.560	*0.5125*
NG	***0.555***	0.505	0.590	*0.535*	0.5463
LEN	0.525	*0.435*	*0.530*	0.560	*0.5125*
LSS+NE	0.525	0.645	0.805	0.585	0.6400
LSS+NE+DEP	0.520	0.650	***0.810***	0.580	0.6400
LSS+NE+TK	0.530	0.625	0.805	0.595	0.6388
LSS+NE+TK+LEN	0.530	0.630	0.805	0.625	0.6475
LSS+NE+TK+DEP	0.530	0.625	0.805	0.620	0.6450
LSS+NE+TK+DEP+NG	*0.460*	0.625	0.785	***0.655***	0.6313
LSS+NE+TK+LEN+DEP	0.525	0.615	0.790	0.600	0.6325
LSS+NE+TK+LEN+DIST (run2)	0.530	0.635	0.805	0.625	***0.6488***
All Features	0.500	0.590	0.790	0.630	0.6275

Table 4. Accuracy by task and selected feature set on the RTE-3 Test dataset (Trained on the RTE-3 development dataset).

4.4 Discussions

4.4.1 Feature Analysis

Table 4 lays out the results of using various feature combinations to train the classifier. All of the models were trained on the RTE 3 development dataset only.

It is obvious that the LSS and NE features have the most utility. The DIST and LEN features seem useless for this dataset, as these features themselves can not beat the baseline system with accuracy 0.545. Systems with individual features perform similarly on SUM pairs except NG, and on IE pairs except NG and DEP features. However, on IR and QA pairs, they behave quite differently. For example, system with NE feature achieves accuracy 0.78 on QA pairs, while system with DEP feature obtains 0.575. NE and LSS features have similar effects, but NE is more useful for QA pairs.

It is interesting to note that some features improve the score in some combinations, but in others they decrease it. For instance, although DEP scores above the baseline at 0.5663, when added to the combination of LSS, NE, TK, and LEN it lowers the overall accuracy by 1.5%.

4.4.2 About Average Precision Measure

As we mentioned in section 4.2, Average Precision (AvgP) is expected to evaluate the ranking ability of a system according to confidence values. However, we found that the current evaluation process and the measure itself have some problems and need to be modified for RTE evaluation.

On one hand, the current evaluation process doesn't consider tied cases where many pairs may have the same confidence value. It is reasonable to assume that the order of tied pairs will be random. Accordingly, the derived Average Precision will vary.

Let's look at a simple example: suppose we have two pairs *c* and *d*, and *c* is the only one positive entailment pair. Here, R=1, N=2 for Equation (3). Two systems X and Y output ranked results as $\{c, d\}$ and $\{d,c\}$ respectively. According to Equation (3), the AvgP value of system X is 1, where that of system Y is 0.5. If these two systems assign same confidence value for both pairs, we can not conclude that system X is better than system Y.

To avoid this problem, we suggest requiring that each system for ranked submission output its confidence for each pair. Then, when calculating Average Precision measure, we first re-rank the list with these confidence values and true answers for each pair. For tied pairs, we rank pairs with true answer "NO" before those with positive entailment relation. By this way, we can produce a stable and more reasonable Average Precision value. For example, in the above example, the modified average precisions for both systems will be 0.5.

On the other hand, from the Equation (3), we know that the upper bound of Average Precision is 1. At the same time, we can also derive a lower bound for this measure as in Equation (4). It corresponds to the worst system which places all the negative pairs before all the positive pairs. The lower bound of Average Precision for RTE-3 test dataset is 0.3172.

$$LB\_AvgP = \frac{1}{R}\sum_{j=0}^{R-1}\frac{R-j}{N-j} . \qquad (4)$$

As the values of N and R change, the lower bound of Average Precision will vary. Therefore, the original Average Precision measure as in Equation (3) is not an ideal one for comparison across datasets.

To solve this problem, we propose a normalized Average Precision measure as in Equation (5).

$$Norm\_AvgP = \frac{AvgP - LB\_AvgP}{1 - LB\_AvgP} . \qquad (5)$$

5 Conclusion and Future Work

In this paper, we report our RTE-3 system. The system was built on a machine learning framework with features produced by state-of-the-art NLP techniques. Lexical semantic similarity and Named entities are the two most effective features. Data analysis shows a higher baseline performance for RTE-3 than RTE-1 and RTE-2, and the current Average Precision measure needs to be changed.

As *T-H* pairs from IE task are the most difficult ones, we will focus on these pairs in our future research.

References

Rod Adams. 2006. Textual Entailment Through Extended Lexical Overlap. In *Proceedings of RTE-2 Workshop*.

Satanjeev Banerjee and Ted Pedersen. 2002. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In *Proceedings of CICLING-02*.

Roy Bar-Haim et al. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of RTE-2 Workshop.*

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and Log-Linear Models. In *Proceedings of ACL-04*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In Quiñonero-Candela et al. (editors.), *MLCW 2005, LNAI Volume 3944*.

Christiane Fellbaum. 1998. *WordNet: an Electronic Lexical Database*. MIT Press.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. *In Proceedings of AAAI-06*.

Ted Pedersen et al. 2005. *Maximizing Semantic Relatedness to Perform Word Sense Disambiguation.* Research Report UMSI 2005/25, Supercomputing Institute, University of Minnesota.

Dragomir Radev, Sasha Blair-Goldensohn, and ZhuZhang. 2001. Experiments in single and multidocument summarization using MEAD. In *Proceedings of DUC 2001*.

Satoshi Sekine. 2002. Manual of Oak System (version 0.1). Computer Science Department, New York University, *http://nlp.cs.nyu.edu/oak*.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco.