Learning Textual Entailment using SVMs and String Similarity Measures

Prodromos Malakasiotis and **Ion Androutsopoulos**
Department of Informatics
Athens University of Economics and Business
Patision 76, GR-104 34 Athens, Greece

Abstract

We present the system that we submitted to the 3rd Pascal Recognizing Textual Entailment Challenge. It uses four Support Vector Machines, one for each subtask of the challenge, with features that correspond to string similarity measures operating at the lexical and shallow syntactic level.

1 Introduction

Textual Entailment is desirable in many natural language processing areas, such as question answering, information extraction, information retrieval, and multi-document summarization. In the Pascal Recognizing Textual Entailment Challenge (RTE), it is defined as the task of deciding whether or not the meaning of a *hypothesis* text (H) can be inferred from the meaning of another text (T).[1] For instance:

T: The drugs that slow down or halt Alzheimer's disease work best the earlier you administer them.

H: Alzheimer's disease is treated using drugs.

is a correct entailment pair, but the following is not:

T: Drew Walker, NHS Tayside's public health director, said: "It is important to stress that this is not a confirmed case of rabies."

H: A case of rabies was confirmed.

In previous RTE challenges (Dagan et al., 2006; Bar-Haim et al., 2006), several machine-learning approaches appeared, but their results showed that significant improvements were still necessary. In this paper, we present the system we used in the third RTE challenge. The latter had four different development and test sets (QA, IR, IE, SUM), intended to evaluate textual entailment recognition in the four natural language processing areas mentioned above.

2 System overview

Our system uses SVMs (Vapnik, 1998) to determine whether each T–H pair constitutes a correct textual entailment or not. In particular, it employs four SVMs, each trained on the development dataset of the corresponding RTE subtask (QA, IR, IE, SUM) and used on the corresponding test dataset. Preliminary experiments indicated that training a single SVM on all four subsets leads to worse results, despite the increased size of the training set, presumably because of differences in how the pairs were constructed in each subtask, which do not allow a single SVM to generalize well over all four.

The system is based on the assumption that string similarity at the lexical and shallow syntactic level can be used to identify textual entailment reasonably well, at least in question answering, the main area we are interested in. We, therefore, try to capture different kinds of similarity by employing 10 different string similarity measures, to be discussed below. In each T–H case, every measure is applied to the following 8 pairs of strings, producing a total of 80 measurements:

pair 1: two strings with the *original words* of T and H, respectively; although we refer to 'words', this and the following string pairs also contain non-word tokens, such as punctuation.[2]

[1] See http://www.pascal-network.org/.

[2] We use OPENNLP's tokenizer, POS-tagger, and chunker (see http://opennlp.sourceforge.net/), and our own implementation of Porter's stemmer.

Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 42–47,
Prague, June 2007.

pair 2: two strings containing the corresponding *stems* of the words of T and H, respectively;

pair 3: two strings containing the *part-of-speech* (POS) tags of the words of T and H;

pair 4: two strings containing the *chunk* tags (see below) of the words of T and H;

pair 5: two strings containing only the *nouns* of T and H, as identified by a POS-tagger;

pair 6: two strings containing only the *stems of the nouns* of T and H;

pair 7: two strings containing only the *verbs* of T and H, as identified by a POS-tagger;

pair 8: two strings containing only the *stems of the verbs* of T and H.

Chunk tags are of the form B-x, I-x or O, were B and I indicate the initial and other words of the chunks, respectively, whereas O indicates words outside all chunks; x can be NP, VP, or PP, for noun phrase, verb phrase, and prepositional phrase chunks.

Partial matches: When applying the string similarity measures, one problem is that T may be much longer than H, or vice versa. Consider, for example, the following T–H pair. The difference in the lengths of T and H may mislead many similarity measures to indicate that the two texts are very dissimilar, even though H is included verbatim in T.

T: Charles de Gaulle died in 1970 at the age of eighty. He was thus fifty years old when, as an unknown officer recently promoted to the (temporary) rank of brigadier general, he made his famous broadcast from London rejecting the capitulation of France to the Nazis after the debacle of May-June 1940.

H: Charles de Gaulle died in 1970.

To address this problem, when we consider a pair of strings (s_1, s_2), if s_1 is longer than s_2, we also compute the ten values $f_i(s'_1, s_2)$, where f_i $(1 \leq i \leq 10)$ are the string similarity measures, for every s'_1 that is a substring of s_1 of the same length as s_2. We then locate the s'_1 with the best average similarity to s_2, shown below as s'^{*}_1:

$$s'^{*}_1 = \arg\max_{s'_1} \sum_{i=1}^{10} f_i(s'_1, s_2)$$

and we keep the ten $f_i(s'^{*}_1, s_2)$ values and their average as 11 additional measurements. Similarly, if s_2 is longer than s_1, we keep the ten $f_i(s_1, s'^{*}_2)$ values and their average. This process could be applied to all pairs 1–8 above, but the system we submitted applied it only to pairs 1–4; hence, there is a total of 44 additional measurements in each T–H case.

The 124 measurements discussed above provide 124 candidate numeric features that can be used by the SVMs.[3] To those, we add the following four:

Negation: Two Boolean features, showing if T or H, respectively, contain negation, identified by looking for words like "not", "won't", etc.

Length ratio: This is $\frac{\min(L_T, L_H)}{\max(L_T, L_H)}$, were L_T and L_H are the lengths, in words, of T and H.

Text length: Binary feature showing if the markup of the dataset flags T as 'long' or 'short'.

Hence, there are 128 candidate features in total. From those, we select a different subset for the SVM of each subtask, as will be discussed in following sections. Note that similarity measures have also been used in previous RTE systems as features in machine learning algorithms; see, for example, Kozareva and Montoyo (2006), Newman et al. (2006). However, the results of those systems indicate that improvements are still necessary, and we believe that one possible improvement is the use of more and different similarity measures.

We did not use similarity measures that operate on parse trees or semantic representations, as we are interested in RTE methods that can also be applied to less spoken languages, where reliable parsers, fact extractors, etc. are often difficult to obtain.

2.1 String similarity measures

We now describe the ten string similarity measures that we use.[4] The reader is reminded that the measures are applied to string pairs (s_1, s_2), where s_1 and s_2 derive from T and H, respectively.

Levenshtein distance: This is the minimum number of operations (edit distance) needed to transform one string (in our case, s_1) into the other one (s_2),

[3] All feature values are normalized in $[-1, 1]$.

[4] We use the SIMMETRICS library; see `http://www.dcs.shef.ac.uk/~sam/simmetrics.html`.

where an operation is an insertion, deletion, or substitution of a single character. In pairs of strings that contain POS or chunk tags, it would be better to consider operations that insert, delete, or substitute entire tags, instead of characters, but the system we submitted did not do this; we addressed this issue in subsequent work, as will be discussed below.

Jaro-Winkler distance: The Jaro-Winkler distance (Winkler, 1999) is a variation of the Jaro distance (Jaro, 1995), which we describe first. The Jaro distance d_j of s_1 and s_2 is defined as:

$$d_j(s_1, s_2) = \frac{m}{3 \cdot l_1} + \frac{m}{3 \cdot l_2} + \frac{m - t}{3 \cdot m},$$

where l_1 and l_2 are the lengths (in characters) of s_1 and s_2, respectively. The value m is the number of characters of s_1 that match characters of s_2. Two characters from s_1 and s_2, respectively, are taken to match if they are identical and the difference in their positions does not exceed $\frac{\max(l_1, l_2)}{2} - 1$. Finally, to compute t ('transpositions'), we remove from s_1 and s_2 all characters that do not have matching characters in the other string, and we count the number of positions in the resulting two strings that do not contain the same character; t is half that number.

The Jaro-Winkler distance d_w emphasizes prefix similarity between the two strings. It is defined as:

$$d_w(s_1, s_2) = d_j(s_1, s_2) + l \cdot p \cdot [1 - d_j(s_1, s_2)],$$

where l is the length of the longest common prefix of s_1 and s_2, and p is a constant scaling factor that also controls the emphasis placed on prefix similarity. The implementation we used considers prefixes up to 6 characters long, and sets $p = 0.1$.

Again, in pairs of strings (s_1, s_2) that contain POS tags or chunk tags, it would be better to apply this measure to the corresponding lists of tags in s_1 and s_2, instead of treating s_1 and s_2 as strings of characters, but the system we submitted did not do this; this issue was also addressed in subsequent work.

Soundex: Soundex is an algorithm intended to map each English name to an alphanumeric code, so that names whose pronunciations are the same are mapped to the same code, despite spelling differences.[5] Although Soundex is intended to be used on names, and in effect considers only the first letter and the first few consonants of each name, we applied it to s_1 and s_2, in an attempt to capture similarity at the beginnings of the two strings; the strings were first stripped of all white spaces and non-letter characters. We then computed similarity between the two resulting codes using the Jaro-Winkler distance. A better approach would be to apply Soundex to all words in T and H, forming a 9th pair (s_1, s_2), on which other distance measures would then be applied; we did this in subsequent work.

Manhattan distance: Also known as City Block distance or L_1, this is defined for any two vectors $\vec{x} = \langle x_1, \ldots, x_n \rangle$ and $\vec{y} = \langle y_1, \ldots, y_n \rangle$ in an n-dimensional vector space as:

$$L_1(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i|.$$

In our case, n is the number of distinct words (or tags) that occur in s_1 and s_2 (in any of the two); and x_i, y_i show how many times each one of these distinct words occurs in s_1 and s_2, respectively.

Euclidean distance: This is defined as follows:

$$L_2(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

In our case, $\vec{x}$ and $\vec{y}$ correspond to s_1 and s_2, respectively, as in the previous measure.

Cosine similarity: The definition follows:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|}.$$

In our system $\vec{x}$ and $\vec{y}$ are as above, except that they are binary, i.e., x_i and y_i are 1 or 0, depending on whether or not the corresponding word (or tag) occurs in s_1 or s_2, respectively.

N-gram distance: This is the same as L_1, but instead of words we use all the (distinct) character n-grams in s_1 and s_2; we used $n = 3$.

Matching coefficient: This is $|X \cap Y|$, where X and Y are the sets of (unique) words (or tags) of s_1 and s_2, respectively; i.e., it counts how many common words s_1 and s_2 have.

[5] See http://en.wikipedia.org/wiki/Soundex.

Dice coefficient: This is the following quantity; in our case, X and Y are as in the previous measure.

$$\frac{2 \cdot |X \cap Y|}{|X| + |Y|}$$

Jaccard coefficient: This is defined as $\frac{|X \cap Y|}{|X \cup Y|}$; again X and Y are as in the matching coefficient.

2.2 SVM tuning and feature selection

As already noted, we employed four SVMs, one for each subtask of the challenge (IR, IE, QA, SUM).[6] In each subtask, feature selection was performed as follows. We started with a set of 20 features, which correspond to the ten similarity measures applied to both words and stems (string pairs 1 and 2 of section 1); see table 1. We then added the 10 features that correspond to the ten similarity measures applied to POS tags (string pair 3). In IE and IR, this addition led to improved leave-one-out cross-validation results on the corresponding development sets, and we kept the additional features (denoted by 'X' in table 1). In contrast, in QA and SUM the additional 10 features were discarded, because they led to no improvement in the cross-validation. We then added the 10 features that corresponded to the ten similarity measures applied to chunk tags (string pair 4), which were retained only in the IE SVM, and so on.

The order in which we considered the various extensions of the feature sets is the same as the order of the rows of table 1, and it reflects the order in which it occurred to us to consider the corresponding additional features while preparing for the challenge. We hope to investigate additional feature selection schemes in further work; for instance, start with all 128 features and explore if pruning any groups of features improves the cross-validation results.

With each feature set that we considered, we actually performed multiple leave-one-out cross-validations on the development dataset, for different values of the parameters of the SVM and kernel, using a grid-search utility. Each feature set was evaluated by considering its best cross-validation result. The best cross-validation results for the final feature sets of the four SVMs are shown in table 2.

[6]We use LIBSVM (Chang and Lin, 2001), with a Radial Basis Function kernel, including LIBSVM's grid search tuning utility.

Subtask	*Accuracy (%)*
QA	86.50 (90.00)
IR	80.00 (75.50)
SUM	73.00 (72.50)
IE	62.00 (61.50)
all	75.38 (74.88)

Table 2: Best cross-validation results of our system on the *development datasets*. Results with subsequent improvements are shown in brackets.

Subtask	*Accuracy (%)*	*Average Precision (%)*
QA	73.50 (76.00)	81.03 (81.08)
IR	64.50 (63.50)	63.61 (67.28)
SUM	57.00 (60.50)	60.88 (61.58)
IE	52.00 (49.50)	58.16 (51.57)
all	61.75 (62.38)	68.08 (68.28)

Table 3: *Official results* of our system. Results with subsequent improvements are shown in brackets.

3 Official results and discussion

We submitted only one run to the third RTE challenge. The official results of our system are shown in table 3.[7] They are worse than the best results we had obtained in the cross-validations on the development datasets (cf. table 2), but this was expected to a large extent, since the SVMs were tuned on the development datasets; to some extent, the lower official results may also be due to different types of entailment being present in the test datasets, which had not been encountered in the training sets.

As in the cross-validation results, our system performed best in the QA subtask; the second and third best results of our system were obtained in IR and SUM, while the worst results were obtained in IE. Although a more thorough investigation is necessary to account fully for these results, it appears that they support our initial assumption that string similarity at the lexical and shallow syntactic level can be used to identify textual entailment reasonably well in question answering systems. Some further reflections on the results of our system follow.

In the QA subtask of the challenge, it appears that each T was a snippet returned by a question answering system for a particular question.[8] We are not aware of exactly how the Ts were selected by the

[7]See the RTE Web site for a definition of 'average precision'.

[8]Consult `http://www.pascal-network.org/Challenges/RTE3/Introduction/`.

Feature sets	*features*	IE	IR	QA	SUM
similarity measures on **words**	10	X	X	X	X
similarity measures on **stems**	10	X	X	X	X
+ similarity measures on **POS tags**	+10	X	X		
+ similarity measures on **chunk tags**	+10	X			X
+ **average** of sim. measures on **words** of best **partial** match	+1				X
+ **average** of sim. measures on **stems** of best **partial** match	+1			X	X
+ **average** of sim. measures on **POS tags** of best **partial** match	+1			X	X
+ **average** of sim. measures on **chunk tags** of best **partial** match	+1		X		X
+ similarity measures on **words** of best **partial** match	+10				
+ similarity measures on **stems** of best **partial** match	+10				X
+ similarity measures on **POS tags** of best **partial** match	+10	X			
+ similarity measures on **chunk tags** of best **partial** match	+10				
+ **negation**	+2	X			
+ **length ratio**	+1	X			
+ similarity measures on **nouns**	+10	X			
+ similarity measures on **noun stems**	+10				
+ similarity measures on **verbs**	+10				X
+ similarity measures on **verb stems**	+10				
+ **short/long** T	+1	X		X	
Total	128	64	31	23	54

Table 1: Feature sets considered and chosen in each subtask.

systems used, but QA systems typically return Ts that contain the expected answer type of the input question; for instance, if the question is "When did Charles de Gaulle die?", T will typically contain a temporal expression. Furthermore, QA systems typically prefer Ts that contain many words of the question, preferably in the same order, etc. (Radev et al., 2000; Ng et al., 2001; Harabagiu et al., 2003). Hence, if the answers are sought in a document collection with high redundancy (e.g., the Web), i.e., a collection where each answer can be found with many different phrasings, the Ts (or parts of them) that most QA systems return are often very similar, in terms of phrasings, to the questions, provided that the required answers exist in the collection.

In the QA datasets of the challenge, for each T, which was a snippet returned by a QA system for a question (e.g., "When did Charle de Gaulle die?"), an H was formed by "plugging into" the question an expression of the expected answer type from T. In effect, this converted all questions to propositions (e.g., "Charle de Gaulle died in 1970.") that require a "yes" or "no" answer. Note that this plugging in does not always produce a true proposition; T may contain multiple expressions of the expected answer type (e.g., "Charle de Gaulle died in 1970. In 1990, a monument was erected…") and the wrong one may be plugged into the question (H = "Charle de Gaulle died in 1990.").

Let us first consider the case where the proposition (H) is true. Assuming that the document collection is redundant and that the answer to the question exists in the collection, T (or part of it) will often be very similar to H, since it will be very similar to the question that H was derived from. In fact, the similarity between T and H may be greater than between T and the question, since an expression from T has been plugged into the question to form H. Being very similar, T will very often entail H, and, hence, the (affirmative) responses of our system, which are based on similarity, will be correct.

Let us now consider the case where H is false. Although the same arguments apply, and, hence, one might again expect T to be very similar to H, this is actually less likely now, because H is false and, hence, it is more difficult to find a very similarly phrased T in the presumed trustful document collection. The reduced similarity between T and H will lead the similarity measures to suggest that the T–H entailment does not hold; and in most cases, this is a correct decision, because H is false and, thus, it cannot be entailed by a (true) T that has been extracted from a trustful document collection.

Similar arguments apply to the IR subtask, where our system achieved its second best results. Our results in this subtask were lower than in the QA subtask, presumably because the Ts were no longer filtered by the additional requirement that they must contain an expression of the expected answer type.

We attribute the further deterioration of our results in the SUM subtask to the fact that, according to the challenge's documentation, all the T–H pairs of that subtask, both true and false entailments, were chosen to have high lexical similarity, which does not allow the similarity measures of our system to distinguish well between the two cases. Finally, the lower results obtained in the IE subtask may be due to the fact that the T–H pairs of that subtask were intended to reflect entailments identified by information extraction systems, which specialize on identifying particular semantic relations by employing more complicated machinery (e.g., named entity recognizers and matchers, fact extractors, etc.) than simple string similarity measures; the results may also be partly due to the four different ways that were used to construct the T–H pairs of that subtask. It is interesting to note (see table 1) that the feature sets were larger in the subtasks where our system scored worse, which may be an indication of the difficulties the corresponding SVMs encountered.

4 Conclusions and further work

We presented a textual entailment recognition system that relies on SVMs whose features correspond to string similarity measures applied to the lexical and shallow syntactic level. Experimental results indicate that the system performs reasonably well in question answering (QA), which was our main target, with results deteriorating as we move to information retrieval (IR), multi-document summarization (SUM), and information extraction (IE).

In work carried out after the official submission of our system, we incorporated two of the possible improvements that were mentioned in previous sections: we treated strings containing POS or chunk tags as lists of tags; and we applied Soundex to each word of T and H, forming a 9th pair of strings, on which all other similarity measures were applied; feature selection was then repeated anew. The corresponding results are shown in brackets in tables 2 and 3. There was an overall improvement in all tasks (QA, IR, SUM), except for IE, where textual entailment is more difficult to capture via textual similarity, as commented above. We have suggested two additional possible improvements: applying partial matching to all of the string pairs that we consider, and investigating other feature selection schemes. In future work, we also plan to exploit WordNet to capture synonyms, hypernyms, etc.

Acknowledgements

This work was funded by the Greek PENED 2003 programme, which is co-funded by the European Union (75%), and the Greek General Secretariat for Research and Technology (25%).

References

R. Bar-Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor. 2006. The 2nd PASCAL recognising textual entailment challenge. In *Proceedings of the 2nd* PASCAL *Challenges Workshop on Recognising Textual Entailment*, Venice, Italy.

C.-C. Chang and C.-J. Lin, 2001. *LIBSVM: a library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

I. Dagan, O. Glickman, and B. Magnini. 2006. The PASCAL recognising textual entailment challenge. In Quiñonero-Candela et al., editor, *MLCW 2005, LNAI*, volume 3904, pages 177–190. Springer-Verlag.

S.M. Harabagiu, S.J. Maiorano, and M.A. Pasca. 2003. Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267.

M.A. Jaro. 1995. Probabilistic linkage of large public health data file. *Statistics in Medicine*, 14:491–498.

Z. Kozareva and A. Montoyo. 2006. MLENT: The machine learning entailment system of the University of Alicante. In *Proc. of 2nd* PASCAL *Challenges Workshop on Recognising Textual Entailment*, Venice, Italy.

E. Newman, J. Dunnion, and J. Carthy. 2006. Constructing a decision tree classifier using lexical and syntactic features. In *Proc. of 2nd* PASCAL *Challenges Workshop on Recognising Textual Entailment*, Venice, Italy.

H.T. Ng, J.L.P. Kwan, and Y. Xia. 2001. Question answering using a large text database: A machine learning approach. In *Proc. of Empirical Methods in Natural Language Processing*, Carnegie Mellon Univ., PA.

D.R. Radev, J. Prager, and V. Samn. 2000. Ranking suspected answers to natural language questions using predictive annotation. In *Proc. of* NAACL-ANLP, pages 150–157, Seattle, WA.

V. Vapnik. 1998. *Statistical learning theory*. John Wiley.

W.E. Winkler. 1999. The state of record linkage and current research problems. Statistical Research Report RR99/04, US Bureau of the Census, Washington, DC.