Paraphrase Recognition Using Machine Learning to Combine Similarity Measures

Prodromos Malakasiotis
Department of Informatics
Athens University of Economics and Business
Patission 76, GR-104 34 Athens, Greece

Abstract

This paper presents three methods that can be used to recognize paraphrases. They all employ string similarity measures applied to shallow abstractions of the input sentences, and a Maximum Entropy classifier to learn how to combine the resulting features. Two of the methods also exploit WordNet to detect synonyms and one of them also exploits a dependency parser. We experiment on two datasets, the MSR paraphrasing corpus and a dataset that we automatically created from the MTC corpus. Our system achieves state of the art or better results.

1 Introduction

Recognizing or generating semantically equivalent phrases is of significant importance in many natural language applications. In question answering, for example, a question may be phrased differently than in a document collection (e.g., "Who *is the author of* War and Peace?" vs. "Leo Tolstoy *is the writer of* War and Peace."), and taking such variations into account can improve system performance significantly (Harabagiu et al., 2003; Harabagiu and Hickl, 2006). A paraphrase generator, meaning a module that produces new phrases or patterns that are semantically equivalent (or almost equivalant) to a given input phrase or pattern (e.g., "X is the writer of Y" $\Leftrightarrow$ "X wrote Y" $\Leftrightarrow$ "Y was written by X" $\Leftrightarrow$ "X is the author of Y", or "X produces Y" $\Leftrightarrow$ "X manufactures Y" $\Leftrightarrow$ "X is the manufacturer of Y") can be used to produce alternative phrasings of the question, before matching it against a document collection.

Unlike paraphrase generators, paraphrase recognizers decide whether or not two *given* phrases (or patterns) are paraphrases, possibly by generalizing over many different training pairs of phrases. Paraphrase recognizers can be embedded in paraphrase generators to filter out erroneous generated paraphrases; but they are also useful on their own. In question answering, for example, they can be used to check if a pattern extracted from the question (possibly by replacing named entities by their semantic categories and turning the question into a statement) matches any patterns extracted from candidate answers. As a further example, in text summarization, especially multi-document summarization, a paraphrase recognizer can be used to check if a sentence is a paraphrase of any other sentence already present in a partially constructed summary.

Note that, although "paraphrasing" and "textual entailment" are sometimes used as synonyms, we use the former to refer to methods that generate or recognize semantically equivalent (or almost equivalent) phrases or patterns, whereas in textual entailment (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007) the expressions or patterns are not necessarily semantically equivalent; it suffices if one entails the other, even if the reverse direction does not hold. For example, "Y was written by X" textually entails "Y is the work of X", but the reverse direction does not necessarily hold (e.g., if Y is a statue); hence, the two sentences are not paraphrases.

In this paper, we focus on paraphrase recognition. We propose three methods that employ string similarity measures, which are applied to several abstractions of a pair of input phrases (e.g., the phrases themselves, their stems, POS tags). The scores returned by the similarity measures are used as features in a Maximum Entropy (ME) classifier (Jaynes, 1957; Good, 1963), which learns to separate true paraphrase pairs from false ones. Two of our methods also exploit WordNet to detect synonyms, and one of them uses additional features to measure similarities of grammatical relations obtained by a dependency parser.[1] Our experiments were conducted on two datasets: the publicly available Microsoft Research Paraphrasing corpus (Dolan et al., 2004) and a dataset that we constructed from the MTC corpus.[2] The experimental results show that our methods perform very well. Even the simplest one manages to achieve state of the art results, even though it uses fewer linguistic resources than other reported systems. The other two, more elaborate methods perform even better.

Section 2 presents the three methods, and section 3 our experiments. Section 4 covers related work. Section 5 concludes and proposes further work.

2 The three methods

The main idea underlying our methods is that by capturing similarities at various shallow abstractions of the input (e.g., the original sentences, the stems of their words, their POS tags), we can recognize paraphrases and textual entailment reasonably well, provided that we learn to assign appropriate weights to the resulting features. Further improvements are possible by recognizing synonyms and by employing similarity measures that operate on the output of dependency grammar parsers.

2.1 Method 1 (INIT)

During training, the first method, called INIT, is given a set $\{\langle S_{1,1}, S_{1,2}, y_1\rangle, \dots, \langle S_{n,1}, S_{n,2}, y_n\rangle\}$, where $S_{i,1}$ and $S_{i,2}$ are sentences (more generally, phrases), $y_i = 1$ (positive class) if the two sentences are paraphrases, and $y_i = -1$ (negative class) otherwise. Each pair of sentences $\langle S_{i,1}, S_{i,2}\rangle$ is converted to a feature vector $\vec{v_i}$, whose values are scores returned by similarity measures that indicate how similar $S_{i,1}$ and $S_{i,2}$ are at various levels of abstraction. The vectors and the corresponding categories $\{\langle \vec{v_1}, y_i\rangle, \dots, \langle \vec{v_n}, y_n\rangle\}$ are given as input to the ME classifier, which learns how to classify new vectors $\vec{v}$, corresponding to unseen pairs of sentences $\langle S_1, S_2\rangle$.

We use nine string similarity measures: Levenshtein distance (edit distance), Jaro-Winkler distance, Manhattan distance, Euclidean distance, cosine similarity, n-gram distance (with $n = 3$), matching coefficient, Dice coefficient, and Jaccard coefficient. To save space, we do not repeat the definitions of the similarity measures here, since they are readily available in the literature and they are also summarized in our previous work (Malakasiotis and Androutsopoulos, 2007).

For each pair of input strings $\langle S_1, S_2\rangle$, we form ten new pairs of strings $\langle s_1^1, s_2^1\rangle, \dots, \langle s_1^{10}, s_2^{10}\rangle$ corresponding to ten different levels of abstraction of S_1 and S_2, and we apply the nine similarity measures to the ten new pairs, resulting in a total of 90 measurements. These measurements are then included as features in the vector $\vec{v}$ that corresponds to $\langle S_1, S_2\rangle$. The $\langle s_1^i, s_2^i\rangle$ pairs are:

$\langle s_1^1, s_2^1\rangle$: two strings consisting of the *original tokens* of S_1 and S_2, respectively, with the original order of the tokens maintained;[3]

$\langle s_1^2, s_2^2\rangle$: as in the previous case, but now the tokens are replaced by their *stems*;

$\langle s_1^3, s_2^3\rangle$: as in the previous case, but now the tokens are replaced by their *part-of-speech* (POS) tags;

$\langle s_1^4, s_2^4\rangle$: as in the previous case, but now the tokens are replaced by their *soundex codes*;[4]

$\langle s_1^5, s_2^5\rangle$: two strings consisting of only the *nouns* of S_1 and S_2, as identified by a POS-tagger, with the original order of the nouns maintained;

$\langle s_1^6, s_2^6\rangle$: as in the previous case, but now with *nouns replaced by their stems*;

$\langle s_1^7, s_2^7\rangle$: as in the previous case, but now with *nouns replaced by their soundex codes*;

$\langle s_1^8, s_2^8\rangle$: two strings consisting of only the *verbs* of S_1 and S_2, as identified by a POS-tagger, with the original order of the verbs maintained;

$\langle s_1^9, s_2^9\rangle$: as in the previous case, but now with *verbs replaced by their stems*;

$\langle s_1^{10}, s_2^{10}\rangle$: as in the previous case, but now with *verbs replaced by their soundex codes.*

Note that the similarities are measured in terms of tokens, not characters. For instance, the edit distance of S_1 and S_2 is the minimum number of operations needed to transform S_1 to S_2, where an operation is an insertion, deletion or substitution of a single token. Moreover, we use high-level

[1]We use Stanford University's ME classifier and parser; see http://nlp.stanford.edu/.

[2]The corpus is available by the LDC, Catalogue Number LDC2002T01, ISBN 1-58563-217-1.

[3]We use Stanford University's tokenizer and POS-tagger, and Porter's stemmer.

[4]Soundex is an algorithm intended to map English names to alphanumeric codes, so that names with the same pronunciations receive the same codes, despite spelling differences; see http://en.wikipedia.org/wiki/Soundex.

POS tags only, i.e., we do not consider the number of nouns, the voice of verbs etc.; this increases the similarity of positive $\langle s_1^3, s_2^3 \rangle$ pairs.

A common problem is that the string similarity measures may be misled by differences in the lengths of S_1 and S_2. This is illustrated in the following examples, where the underlined part of S_1 is much more similar to S_2 than the entire S_1.

S_1: While Bolton apparently fell and was immobilized, Selenski used the mattress to scale a 10-foot, razor-wire fence, Fischi said.

S_2: After the other inmate fell, Selenski used the mattress to scale a 10-foot, razor-wire fence, Fischi said.

To address this problem, when we consider a pair of strings $\langle s_1, s_2 \rangle$, if s_1 is longer than s_2, we obtain all of the substrings s_1' of s_1 that have the same length as s_2. Then, for each s_1', we compute the nine values $f_j(s_1', s_2)$, where f_j $(1 \leq j \leq 9)$ are the string similarity measures. Finally, we locate the s_1' with the best average similarity (over all similarity measures) to s_2, namely $s_1'^*$:

$$s_1'^* = \arg\max_{s_1'} \sum_{j=1}^{10} f_j(s_1', s_2)$$

and we keep the nine $f_j(s_1'^*, s_2)$ values and their average as ten additional measurements. Similarly, if s_2 is longer than s_1, we keep the nine $f_j(s_1, s_2'^*)$ values and their average. This process is applied to pairs $\langle s_1^1, s_2^1 \rangle, \ldots, \langle s_1^4, s_2^4 \rangle$, where large length differences are more likely to appear, adding 40 more measurements (features) to the vector $\vec{v}$ of each $\langle S_1, S_2 \rangle$ pair of input strings.

The measurements discussed above provide 130 numeric features.[5] To those, we add two Boolean features indicating the existence or absence of negation in S_1 or S_2, respectively; negation is detected by looking for words like "not", "won't" etc. Finally, we add a length ratio feature, defined as $\frac{\min(L_{S_1}, L_{S_2})}{\max(L_{S_1}, L_{S_2})}$, where L_{S_1} and L_{S_2} are the lengths, in tokens, of S_1 and S_2. Hence, there is a total of 133 available features in INIT.

2.2 Method 2 (INIT+WN)

Paraphrasing may involve using synonyms which cannot be detected by the features we have considered so far. In the following pair of sentences, for example, "dispatched" is used as a synonym of "sent"; treating the two verbs as the same token during the calculation of the string similarity measures would yield a higher similarity. The second method, called INIT+WN, treats words from S_1 and S_2 that are synonyms as identical; otherwise the method is the same as INIT.

S_1: Fewer than a dozen FBI agents were dispatched to secure and analyze evidence.

S_2: Fewer than a dozen FBI agents will be sent to Iraq to secure and analyze evidence of the bombing.

2.3 Method 3 (INIT+WN+DEP)

The features of the previous two methods operate at the lexical level. The third method, called INIT+WN+DEP, adds features that operate on the grammatical relations (dependencies) a dependency grammar parser returns for S_1 and S_2. We use three measures to calculate similarity at the level of grammatical relations, namely S_1 dependency recall (R_1), S_2 dependency recall (R_2) and their F-measure (F_{R_1,R_2}), defined below:

$$R_1 = \frac{|common\ dependencies|}{|S_1\ dependencies|}$$

$$R_2 = \frac{|common\ dependencies|}{|S_2\ dependencies|}$$

$$F_{R_1,R_2} = \frac{2 \cdot R_1 \cdot R_2}{R_1 + R_2}$$

The following two examples illustrate the usefulness of dependency similarity measures in detecting paraphrases. In the first example S_1 and S_2 are not paraphrases and the scores are low, while in the second example where S_1 and S_2 have almost identical meanings, the scores are much higher. Figures 1 and 2 lists the grammatical relations (dependencies) of the two sentences with the common ones shown in bold.

Example 1:

S_1: Gyorgy Heizler, head of the local disaster unit, said the coach was carrying 38 passengers.

S_2: The head of the local disaster unit, Gyorgy Heizler, said the coach driver had failed to heed red stop lights.

$R_1 = 0.43$, $R_2 = 0.32$, $F_{R_1,R_2} = 0.36$

Example 2:

S_1: Amrozi accused his brother, whom he called "the witness", of deliberately distorting his evidence.

S_2: Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence.

$R_1 = 0.69$, $R_2 = 0.6$, $F_{R_1,R_2} = 0.64$

[5]All feature values are normalized in $[-1, 1]$. We use our own implementation of the string similarity measures.

Grammatical relations of S_1

mod(Heizler-2, Gyorgy-1)
arg(said-11, Heizler-2)
mod(Heizler-2, head-4)
mod(head-4, of-5)
mod(unit-9, the-6)
mod(unit-9, local-7)
mod(unit-9, disaster-8)
arg(of-5, unit-9)
mod(coach-13, the-12)
arg(carrying-15, coach-13)
aux(carrying-15, was-14)
arg(said-11, carrying-15)
mod(passengers-17, 38-16)
arg(carrying-15, passengers-17)

Grammatical relations of S_2

mod(head-2, The-1)
arg(said-12, head-2)
mod(head-2, of-3)
mod(unit-7, the-4)
mod(unit-7, local-5)
mod(unit-7, disaster-6)
arg(of-3, unit-7)
mod(Heizler-10, Gyorgy-9)
mod(unit-7, Heizler-10)
mod(driver-15, the-13)
mod(driver-15, coach-14)
arg(failed-17, driver-15)
aux(failed-17, had-16)
arg(said-12, failed-17)
aux(heed-19, to-18)
arg(failed-17, heed-19)
mod(lights-22, red-20)
mod(lights-22, stop-21)
arg(heed-19, lights-22)

Figure 1: Grammatical relations of example 1.

Grammatical relations of S_1

arg(accused-2, Amrozi-1)
mod(brother-4, his-3)
arg(accused-2, brother-4)
arg(called-8, whom-6)
arg(called-8, he-7)
mod(brother-4, called-8)
mod(witness-11, the-10)
dep(called-8, witness-11)
mod(brother-4, of-14)
mod(distorting-16, deliberately-15)
arg(of-14, distorting-16)
mod(evidence-18, his-17)
arg(distorting-16, evidence-18)

Grammatical relations of S_2

dep(accused-12, Referring-1)
mod(Referring-1, to-2)
arg(to-2, him-3)
cc(him-3, as-4)
dep(as-4, only-5)
mod(witness-8, the-7)
conj(him-3, witness-8)
arg(accused-12, Amrozi-11)
mod(brother-14, his-13)
arg(accused-12, brother-14)
mod(brother-14, of-15)
mod(distorting-17, deliberately-16)
arg(of-15, distorting-17)
mod(evidence-19, his-18)
arg(distorting-17, evidence-19)

Figure 2: Grammatical relations of example 2.

As with POS-tags, we use only the highest level of the tags of the grammatical relations, which increases the similarity of positive pairs of S_1 and S_2. For the same reason, we ignore the directionality of the dependency arcs which we have found to improve the results. INIT+WN+DEP employs a total of 136 features.

2.4 Feature selection

Larger feature sets do not necessarily lead to improved classification performance. Despite seeming useful, some features may in fact be too noisy or irrelevant, increasing the risk of overfitting the training data. Some features may also be redundant, given other features; thus, feature selection methods that consider the value of each feature on its own (e.g., information gain) may lead to suboptimal feature sets.

Finding the best subset of a set of available features is a search space problem for which several methods have been proposed (Guyon et al., 2006). We have experimented with a wrapper approach, whereby each feature subset is evaluated according to the predictive power of a classifier (treated as a black box) that uses the subset; in our experiments, the predictive power was measured as F-measure (defined below, not to be confused with F_{R_1,R_2}). More precisely, during feature selection, for each feature subset we performed 10-fold cross validation on the training data to evaluate its predictive power. After feature selection, the classifier was trained on all the training data, and it was evaluated on separate test data.

With large feature sets, an exhaustive search over all subsets is intractable. Instead, we experimented with forward hill-climbing and beam search (Guyon et al., 2006). Forward hill-climbing starts with an empty feature set, to which it adds features, one at a time, by preferring to add at each step the feature that leads to the highest predictive power. Forward beam search is similar, except that the search frontier contains the k best examined states (feature subsets) at each time; we used $k = 10$. For $k = 1$, beam search reduces to hill-climbing.

3 Experiments

We now present our experiments, starting from a description of the datasets used.

3.1 Datasets

We mainly used the Microsoft Research (MSR) Paraphrasing Corpus (Dolan et al., 2004), which consists of 5,801 pairs of sentences. Each pair is manually annotated by two human judges as a true or false paraphrase; a third judge resolved disagreements. The data are split into 4,076 training pairs and 1,725 testing pairs.

We have experimented with a dataset we created from the MTC corpus. MTC is a corpus containing news articles in Mandarin Chinese; for each article 11 English translations (by different translators) are also provided. We considered the translations of the same Chinese sentence as paraphrases. We obtained all the possible paraphrase pairs and we added an equal number of randomly selected non paraphrase pairs, which contained sentences that were not translations of the same sentence. In this way, we constructed a dataset containing 82,260 pairs of sentences. The dataset was then split in training (70%) and test (30%) parts, with an equal number of positive and negative pairs in each part.

3.2 Evaluation measures and baseline

We used four evaluation measures, namely accuracy (correctly classified pairs over all pairs), precision (P, pairs correctly classified in the positive class over all pairs classified in the positive class), recall (R, pairs correctly classified in the positive class over all true positive pairs), and F-measure (with equal weight on precision and recall, defined as $\frac{2 \cdot P \cdot R}{P+R}$). These measures are not to be confused with the R_1, R_2, and F_{R_1,R_2} of section 2.3 which are used as features.

A reasonable baseline method (BASE) is to use just the edit distance similarity measure and a threshold in order to decide whether two phrases are paraphrases or not. The threshold is chosen using a grid search utility and 10-fold cross validation on the training data. More precisely, in a first step we search the range [-1, 1] with a step of 0.1.[6] In each step, we perform 10-fold cross validation and the value that achieves the best F-measure is our initial threshold, th, for the second step. In the second step, we perform the same procedure in the range [th - 0.1, th + 0.1] and with a step of 0.001.

[6]Recall that we normalize similarity in [-1, 1].

3.3 Experimental results

With both datasets, we experimented with a Maximum Entropy (ME) classifier. However, preliminary results (see table 1) showed that our MTC dataset is very easy. BASE achieves approximately 95% in accuracy and F-measure, and an approximate performance of 99.5% in all measures (accuracy, precision, recall, F-measure) is achieved by using ME and only some of the features of INIT (we use 36 features corresponding to pairs $\langle s_1^1, s_2^1 \rangle$, $\langle s_1^2, s_2^2 \rangle$, $\langle s_1^3, s_2^3 \rangle$, $\langle s_1^4, s_2^4 \rangle$ plus the two negation features). Therefore, we did not experiment with the MTC dataset any further.

Table 2 (upper part) lists the results of our experiments on the MSR corpus. We optionally performed feature selection with both forward hill-climbing (FHC) and forward beam search (FBS). All of our methods clearly perform better than BASE. As one might expect, there is a lot of redundancy in the complete feature set. Hence, the two feature selection methods (FHC and FBS) lead to competitive results with much fewer features (7 and 10, respectively, instead of 136). However, feature selection deteriorates performance, especially accuracy, i.e., the full feature set is better, despite its redundancy. Table 2 also includes all other reported results for the MSR corpus that we are aware of; we are not aware of the exact number of features used by the other researchers.

It is noteworthy that INIT achieves state of the art performance, even though the other approaches use many more linguistic resources. For example, Wan et al.'s approach (Wan et al., 2006), which achieved the best previously reported results, is similar to ours, in that it also trains a classifier with similarity measures; but some of Wan et al.'s measures require a dependency grammar parser, unlike INIT. More precisely, for each pair of sentences, Wan et al. construct a feature vector with values that measure lexical and dependency similarities. The measures are: word overlap, length difference (in words), BLEU (Papineni et al., 2002), dependency relation overlap (i.e., R_1 and R_2 but not F_{R_1,R_2}), and dependency tree edit distance. The measures are also applied on sequences containing the lemmatized words of the original sentences, similarly to one of our levels of abstraction. Interestingly, INIT achieves the same (and slightly better) accuracy as Wan et al.'s system, without employing any parsing. Our more enhanced methods, INIT+WN and INIT+WN+DEP, achieve even better results.

Zhang and Patrick (2005) use a dependency grammar parser to convert passive voice phrases to active voice ones. They also use a preprocessing stage to generalize the pairs of sentences. The preprocessing replaces dates, times, percentages, etc. with generic tags, something that we have also done in the MSR corpus, but it also replaces words and phrases indicating future actions (e.g., "plans to", "be expected to") with the word "will"; the latter is an example of further preprocessing that could be added to our system. After the preprocessing, Zhang and Patrick create for each sentence pair a feature vector whose values measure the lexical similarity between the two sentences; they appear to be using the maximum number of consecutive common words, the number of common words, edit distance (in words), and modified n-gram precision, a measure similar to BLEU. The produced vectors are then used to train a decision tree classifier. Hence, Zhang and Patrick's approach is similar to ours, but we use more and different similarity measures and several levels of abstraction of the two sentences. We also use ME, along with a wrapper approach to feature selection, rather than decision tree induction and its embedded information gain-based feature selection. Furthermore, all of our methods, even INIT which employs no parsing at all, achieve better results compared to Zhang and Patrick's.

Qiu et al. (2006) first convert the sentences into tuples using parsing and semantic role labeling. They then match similar tuples across the two sentences, and use an SVM (Vapnik, 1998) classifier to decide whether or not the tuples that have not been matched are important or not. If not, the sentences are paraphrases. Despite using a parser and a semantic role identifier, Qiu et al.'s system performs worse than our methods.

Finally, Finch et al.'s system (2005) achieved the second best overall results by employing POS tagging, synonymy resolution, and an SVM. Interestingly, the features of the SVM correspond to machine translation evaluation metrics, rather than string similarity measures, unlike our system. We plan to examine further how the features of Finch et al. and other ideas from machine translation can be embedded in our system, although INIT+WN+DEP outperforms Finch et al.'s system. Interestingly, even when not using more resources than Finch et al. as in methods INIT and INIT+WN

method	*features*	*accuracy*	*precision*	*recall*	*F-measure*
BASE	–	95.30	98.16	92.32	95.15
INIT'	38	99.62	99.50	99.75	99.62

Table 1: Results (%) of our methods on our MTC dataset.

method	*features*	*accuracy*	*precision*	*recall*	*F-measure*
BASE	1	69.04	72.42	86.31	78.76
INIT	133	75.19	78.51	86.31	82.23
INIT+WN	133	75.48	78.91	86.14	82.37
INIT+WN+DEP	136	76.17	79.35	86.75	82.88
INIT+WN+DEP + FHC	7	73.86	75.14	90.67	82.18
INIT+WN+DEP + FBS	10	73.68	73.68	93.98	82.61
Finch et al.	–	74.96	76.58	89.80	82.66
Qiu et al.	–	72.00	72.50	93.40	81.60
Wan et al.	–	75.00	77.00	90.00	83.00
Zhang & Patrick	–	71.90	74.30	88.20	80.70

Table 2: Results (%) of our methods (upper part) and other methods (lower part) on the MSR corpus.

we achieve similar or better accuracy results.

4 Related work

We have already made the distinction between paraphrase (and textual entailment) *generators* vs. *recognizers*, and we have pointed out that recognizers can be embedded in generators as filters. The latter is particularly useful in bootstrapping paraphrase generation approaches (Riloff and Jones, 1999; Barzilay and McKeown, 2001; Ravichandran and Hovy, 2001; Ravichandran et al., 2003; Duclaye et al., 2003; Szpektor et al., 2004), which are typically given seed pairs of named entities for which a particular relation holds; the system locates in a document collection (or the entire Web) contexts were the seeds cooccur, and uses the contexts as patterns that can express the relation; the patterns are then used to locate new named entities that satisfy the relation, and a new iteration begins. A paraphrase recognizer could be used to filter out erroneous generated paraphrases between iterations.

Another well known paraphrase generator is Lin and Pantel's (2001) DIRT, which produces slotted semantically equivalent patterns (e.g., "X is the writer of Y" $\Leftrightarrow$ "X wrote Y" $\Leftrightarrow$ "Y was written by X" $\Leftrightarrow$ "X is the author of Y"), based on the assumption that different paths of dependency trees (obtained from a corpus) that occur frequently with the same words (slot fillers) at their ends are often paraphrases. An extension of DIRT, named LEDIR, has also been proposed (Bhagat et al., 2007) to recognize directional textual entailment rules (e.g., "Y was written by X" $\Rightarrow$ "Y is the work of X"). Ibrahim et al.'s (2003) method is similar to DIRT, but it uses only dependency grammar paths from aligned sentences (from a parallel corpus) that share compatible anchors (e.g., identical strings, or entity names of the same semantic category). Shinyama and Sekine (2003) adopt a very similar approach.

In another generation approach, Barzilay and Lee (2002; 2003) look for pairs of slotted word lattices that share many common slot fillers; the lattices are generated by applying a multiple-sequence alignment algorithm to a corpus of multiple news articles about the same events. Finally, Pang et al. (2003) create finite state automata by merging parse trees of aligned sentences from a parallel corpus; in each automaton, different paths represent paraphrases. Again, a paraphrase recognizer could be embedded in all of these methods, to filter out erroneous generated patterns.

5 Conclusions and further work

We have presented three methods (INIT, INIT+WN, INIT+WN+DEP) that recognize paraphrases given pairs of sentences. These methods employ nine string similarity measures applied to ten shallow abstractions of the input sentences. Moreover, INIT+WN and INIT+WN+DEP exploit WordNet for synonymy resolution, and INIT+WN+DEP uses additional features that measure grammatical relation similarity. Supervised machine learning is used to learn how to combine the resulting features. We experimented with a Maximum Entropy classifier on two datasets; the publicly available MSR corpus and one that we constructed from the

MTC corpus. However, the latter was found to be very easy, and consequently we mainly focused on the MSR corpus.

On the MSR corpus, all of our methods achieved similar or better performance than the sate of the art, even INIT, despite the fact that it uses fewer linguistic resources. Hence, INIT may have practical advantages in less spoken languages, which have limited resources. The most elaborate of our methods, INIT+WN+DEP, achieved the best results, but it requires WordNet and a reliable dependency grammar parser. Feature selection experiments indicate that there is significant redundancy in our feature set, though the full feature set leads to better performance than the subsets produced by feature selection. Further improvements may be possible by including in our system additional features, such as BLEU scores or features for word alignment.

Our long-term goal is to embed our recognizer in a bootstrapping paraphrase generator, to filter out erroneous paraphrases between bootstrapping iterations. We hope that our recognizer will be adequate for this purpose, possibly in combination with a human in the loop, who will inspect paraphrases the recognizer is uncertain of.

Acknowledgements

This work was funded by the Greek PENED 2003 programme, which is co-funded by the European Union (80%), and the Greek General Secretariat for Research and Technology (20%).

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.

R. Barzilay and L. Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of* EMNLP, pages 164–171, Philadelphia, PA.

R. Barzilay and L. Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of* HLT-NAACL, pages 16–23, Edmonton, Canada.

R. Barzilay and K. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of* ACL/EACL, pages 50–57, Toulouse, France.

R. Bhagat, P. Pantel, and E. Hovy. 2007. LEDIR: An unsupervised algorithm for learning directionality of inference rules. In *Proceedings of the* EMNLP-CONLL, pages 161–170.

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

B. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of* COLING, page 350, Morristown, NJ.

F. Duclaye, F. Yvon, and O. Collin. 2003. Learning paraphrases to improve a question-answering system. In *Proceedings of the* EACL *Workshop on Natural Language Processing for Question Answering Systems*, pages 35–41, Budapest, Hungary.

A. Finch, Y. S. Hwang, and E. Sumita. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the 3rd International Workshop on Paraphrasing*, Jeju Island, Korea.

D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan. 2007. The third Pascal recognizing textual entailment challenge. In *Proceedings of the* ACL-*Pascal Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague, Czech Republic.

I. J. Good. 1963. Maximum entropy for hypothesis formulation, especially for multidimentional contigency tables. *Annals of Mathematical Statistics*, 34:911–934.

I.M. Guyon, S.R. Gunn, M. Nikravesh, and L. Zadeh, editors. 2006. *Feature Extraction, Foundations and Applications*. Springer.

S. Harabagiu and A. Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of* COLING-ACL, pages 905–912, Sydney, Australia.

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

A. Ibrahim, B. Katz, and J. Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the* ACL *Workshop on Paraphrasing*, pages 57–64, Sapporo, Japan.

E. T. Jaynes. 1957. Information theory and statistical mechanics. *Physical Review*, 106:620–630.

D. Lin and P. Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7:343–360.

P. Malakasiotis and I. Androutsopoulos. 2007. Learning textual entailment using svms and string similarity measures. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47, Prague, June. Association for Computational Linguistics.

B. Pang, K. Knight, and D. Marcu. 2003. Syntax-based alignment of multiple translations: extracting paraphrases and generating new sentences. In *Proceedings of* HLT-NAACL, pages 102–109, Edmonton, Canada.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, Pennsylvania.

L. Qiu, M. Y. Kan, and T.S. Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of* EMNLP, pages 18–26, Sydney, Australia.

D. Ravichandran and E. Hovy. 2001. Learning surface text patterns for a question answering system. In *Proceedings of* ACL, pages 41–47, Philadelphia, PA.

D. Ravichandran, A. Ittycheriah, and S. Roukos. 2003. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of* HLT-NAACL, pages 85–87, Edmonton, Canada.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of* AAAI, pages 474–479, Orlando, FL.

Y. Shinyama and S. Sekine. 2003. Paraphrase acquisition for information extraction. In *Proceedings of the* ACL *Workshop on Paraphrasing*, Sapporo, Japan.

I. Szpektor, H. Tanev, I. Dagan, and B. Coppola. 2004. Scaling Web-based acquisition of entailment relations. In *Proceedings of* EMNLP, Barcelona, Spain.

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

S. Wan, M. Dras, R. Dale, and C. Paris. 2006. Using dependency-based features to take the "parafarce" out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, pages 131–138, Sydney, Australia.

Y. Zhang and J. Patrick. 2005. Paraphrase identification by text canonicalization. In *Proceedings of the Australasian Language Technology Workshop*, pages 160–166, Sydney, Australia.