Dissimilarity Kernels for Paraphrase Identification

Mihai Lintean and **Vasile Rus**
Department of Computer Science
Institute for Intelligent Systems
The University of Memphis
Memphis, TN 38152, USA
mclinten,vrus@memphis.edu

Abstract

We present in this paper a novel solution to the problem of paraphrase identification based on lexical dissimilarity kernels. Lexical kernels in conjunction with Support Vector Machines are preferred over other learning methods, e.g. decision trees, due to their ability to handle a high number of features. Dissimilarity-based kernels emphasize dissimilarities among text fragments and therefore are appropriate for text similarity tasks characterized by high lexical overlap. We conducted experiments with our kernels on the Microsoft Research (MSR) Paraphrase Corpus, a standardized data set used for assessing approaches to paraphrase identification. Our reported accuracy results are competitive and robust when compared to state-of-the-art single-model approaches. The results were obtained using 10-fold cross-validation over the entire corpus. We also report competitive results on the test portion of the MSR Paraphrase Corpus, which is the standard way to report results on this corpus.

Introduction

Assessing the semantic similarity of texts (words, sentences, paragraphs, documents) is an important task in many applications ranging from question answering (Ibrahim, Katz and Lin 2003) to educational systems (Graesser et al. 2005) to automatic detection of duplicate bug reports in software testing (Rus et al. 2009).

The similarity of two texts can be defined quantitatively or qualitatively in the form of semantic relations such as elaboration (Rus et al. 2009), entailment (Dagan, Glickman and Magnini 2005), or paraphrase (Dolan and Brockett 2005). We focus in this paper on the semantic relation of paraphrase between two sentences.

As an example of a paraphrase relation, we show below a pair of sentences from the Microsoft Research (MSR) Paraphrase Corpus (Dolan and Brockett 2005) in which Text A is a paraphrase of Text B and vice versa.

Text A: *York had no problem with MTAs insisting the decision to shift funds had been within its legal rights.*

Text B: *York had no problem with MTAs saying the decision to shift funds was within its powers.*

We propose an approach that assesses the similarity of dissimilarities between the two sentences in a pair. That is, we first represent each instance (two sentences which may be or not in a paraphrase relation) in the dataset as a vector of dissimilarities. There is one dimension in the vector for each word or sequence of words (bigrams) in the input sentences. The corresponding value for each dimension is non-zero only for words and sequences of words that are present in one sentence but not the other. Dimensions which correspond to common words or sequences are zero. A kernel function is then defined on the dissimilarity vectors that computes the degree of similarity between any two vectors, i.e. it quantifies the similarity of the dissimilarities.

There are two major differences between our work reported here and previous efforts on sentential paraphrase identification. *First*, our method relies on lexical kernels defined over words and sequences of words in the input sentences. The kernels are dissimilarity kernels because they focus on dissimilarities between two sentences of an instance. We use the kernels in combination with a dual learning method, i.e. Support Vector Machines (SVMs), to efficiently handle the high-dimensionality of the representation space for the instances. The space has one dimension for each possible word or combination of words. Dual learning methods handle examples by computing dot-products in an efficient way without explicitly iterating over all features. *Second*, we report more robust results obtained using 10-fold cross-validation in addition to using the original training-testing split in the MSR corpus. All previously reported investigations used the original testing subset of the corpus to report results.

The paper is organized as follows. Next section, *Related Work*, discusses previous work in the area of paraphrase identification highlighting major points and differences among these approaches. The *Similarity of Dissimilarities Approach* section describes the basic idea of our approach with an emphasis on dissimilarity kernels. The *Experiments Results* section presents the experimental setup and discusses the results. The paper ends with conclusions and future work.

Related Work

Previous attempts to address the task of paraphrase identification range from simple to sophisticated. An example of a simple, but accurate approach was proposed by Zhang and Patrick (2005) who reported best results when using a small set of word substring overlap features (they used four features) in combination with a decision tree learning method. They also tried additional text canonicalization processing steps, such as converting passive voice sentence to active voice, but the overall accuracy dropped. The simple approach of Zhang and Patrick is slightly better than the best results reported by Mihalcea, Corley, and Strapparava (2006) who used word-to-word similarity metrics based on word distributional information in corpora or knowledge embedded in a lexical database (WordNet; (Pedersen et al. 2004)). Mihalcea and colleagues report best results when using an average over all metrics. It seems that one problem with Mihalcea and colleagues' approach is the greedy strategy used to semantically match a word from one sentence to a word from the other sentence in a pair. They match the word with the maximum similarity among all the words in the second sentence. Fernando and Stevenson (2008) on the other hand used the same basic idea of using word similarity metrics to identify paraphrases but instead of adopting a greedy strategy they opted for a more global strategy in which the similarity of all pairs of words is considered instead of just the maximum similarities. Fernando and Stevenson's approach is called matrix similarity and provided significantly better results than Mihalcea and colleagues in terms of accuracy. However, Mihalcea, Corley, and Strapparava (2006) had better recall results. An example of a sophisticated approach is the one recently proposed by Das and Smith (2009) who use a probabilistic approach that relies on both syntactic and lexical semantics information to decide whether two sentences are paraphrases or not.

From one perspective, the above methods can be classified as similarity-centered, dissimilarity-centered, or a combination of similarity and dissimilarity approaches. The similarity-centered (e.g., Mihalcea, Corley, and Strapparava (2006); Fernando and Stevenson (2008); Das and Smith (2009)) focus on how similar two sentences are, based on which a similarity score is computed which is then used to make a decision: paraphrase or not. On the other hand, other researchers observed that sentential paraphrases in general have a high degree of lexical overlap and thus decided to focus on dissimilarities between sentences in a pair (Qiu et al. 2006). Some others focused on both similarities and dissimilarities (Lintean and Rus 2009; Wan et al. 2006). Our core approach falls within the dissimilarity-centered category as the basic idea is to detect the degree of dissimilarity between the two sentences in a pair. We use this basic idea in combination with Support Vector Machines to build up classifiers from the training data.

Support Vector Machines (SVMs; (Shave-Taylor and Cristianini 2000)) are one class of supervised machine learning methods that can be used for classification and regression tasks. SVMs search for a separation hyperplane that divides the training instances, represented as points in a multi-dimensional space, into classes. SVMs aim at finding a hyperplane that simultaneously minimizes the empirical classification error and maximizes the geometric margin between the hyperplane and the nearest data points. Maximum-margin hyperplanes lead in general to smaller generalization errors, a desirable outcome. SVMs rely on a kernel function and solving a quadratic programming optimization problem with linear constraints for which techniques and methods exist. Due to its ability to work with highly-dimensional spaces, SVMs are a good fit for tackling natural language processing problems such as the one we address in this paper. Of particular interest was the definition of new kernel functions that range from substring kernels that compute a weighted-sum of common substrings in two documents in order to classify them (Lohdi et al. 2008) to dependency kernels that compute dependency substring overlap (Kate 2008). Along the same lines, we define our own kernel that quantifies the degree of dissimilarity among two sentences given with each instance of our dataset. Kernelized SVMs have been previously studied by (Zanzotto, Pannacchiotti and Moschitti 2009) for the task of recognizing textual entailnment. Their method differs from ours in that they look at similarities between the syntactic trees of the paired sentences. To the best of our knowledge, no one has approached the paraphrase identification task with a newly defined kernel function although standard kernels have been used. For instance, Wan et al. (2006) have used a small set of expert-defined features for the input space together with a polynomial kernel available in WEKA, a machine learning toolkit.

Similarity of Dissimilarities Approach

Our approach is based on representing each instance, i.e. pair of sentences, in a vectorial representation in which there is a dimension for each word or sequences of words in the data. Actually, dimensions correspond to word types[1] or sequences of word types. We only experimented with unigrams and bigrams. Bigrams can be considered equivalent to using word order information. As Collins (1996) noted, 70% of the dependencies in English are between adjacent words, which mean bigrams can capture many syntactic dependencies. The value along each dimension in this vectorial representation is zero if the corresponding word-type does not occur in any of the paired sentences, or occurs in both sentences. If a word type or sequence is present in a sentence but not the other, then the value is non-zero. Non-zero values can represent weights that measure the importance of the dimension/word-type for a particular instance. Various weighting schemes are possible: binary, raw frequency, idf (Dumais 1991) or entropy. We report results using binary and raw frequency weights, leaving the other schemes for future studies. Based on this representation, a kernel is then defined that efficiently computes the similarities between these vectors. That is, two instances that have similar dissimilarities should be projected close to each other in the feature space generated by the kernel.

Kernels

The role of a kernel in Support Vector Machines is to map the initial input space into a new space in which a linear hyperplane could separate the instances. As long as one can find an efficient way to compute the kernel, there is no need to operate in the newly projected feature space. According

[1]A word type is an unique word in the input texts

to Mercer's Theorem (Shave-Taylor and Cristianini 2000), any semi-positive definite, symmetric function can serve as a kernel. One simple SVMs kernel for our problem and vector representation is to count the number of common words or word sequences of two given instances. Such kernels are a generalization from string to word sequences of the string kernel proposed by Lodhi et al. (2008).

For the paraphrase identification problem, given that one instance is represented by a pair of two sentences, we will first derive the vector representation described earlier and then define a kernel that computes the number of similarities among these vectors. As the vectors encode dissimilarities, we actually compute the degree of two instances having the same kind of dissimilarities between the sentences of the instance. That is, the assumption is that *instances which have many common dissimilarities should be very similar* or next to each other in the projected feature space.

We define the kernel value between two instances A and B of paired sentences (S_{A1}, S_{A2}) and (S_{B1}, S_{B2}) as in Equation 1. We represent sentences by their corresponding set of n-grams (unigrams or bigrams; punctuation can also be included here). We denote with $S_{A1}\Delta S_{A2}$ the symmetric difference between the two sets, meaning the n-grams present in S_{A1} or S_{A2} but not both, while $w_1 \equiv w_2$ means identical n-grams. The $weight$ function retrieves the weight for the corresponding n-gram. We use two weights, binary and raw frequency of types, in the reported experiments.

$$K(A,B) = \sum_{\substack{w_1 \in S_{A1}\Delta S_{A2} \\ w_2 \in S_{B1}\Delta S_{B2} \\ w_1 \equiv w_2}} weight(w_1) * weight(w_2) \quad (1)$$

Although equation 1 suggests, in worst case scenario, a quadratic time complexity in the length of all four sentences, we can achieve linear complexity on the number of differences that were found in the paired sentences. First, to detect the n-grams that are different between the two paired sentences in each instance, we can have quadratic complexity, if a simple method of comparing all n-grams is used, or achieve an even better complexity of $O(nlog(n))$, if we do a quick-sort on the n-grams before comparing them. However, this can be done initially on each instance, before applying the SVM classifier, and making sure that, after the comparison, the detected differences are lexically sorted. As a result, the instances will be represented by sorted lists of n-gram. Then, the complexity of our kernel function is reduced to a simple linear comparison on two sorted lists.

Therefore, a big advantage of the proposed kernel is its linear time complexity. A second advantage would be the ease of interpretation by humans as the dimensions correspond to words or word sequences.

Experiments and Results

We experimented with our approach on the MSR Paraphrase Corpus (Dolan and Brockett 2005). The MSR Paraphrase Corpus is the largest publicly available annotated paraphrase corpus which has been used in most of the recent studies that addressed the problem of paraphrase identification. The corpus consists of 5801 sentence pairs collected from newswire articles, 3900 of which were labeled as paraphrases by human annotators. The whole set is divided into a training subset (4076 sentences of which 2753 are true paraphrases) which we have used to determine the optimum threshold T, and a test subset (1725 pairs of which 1147 are true paraphrases) that is used to report the performance results.

There are several critiques about MSR corpus. First, MSR has too much word overlap (spawning form the method used to collect the data set) and less syntactic diversity. Therefore, the corpus cannot be used to learn paraphrase syntactic patterns (Zhang and Patrick 2005; Weeds 2005). Given the high lexical overlap, a good strategy would be to focus on differences among the sentences in a pair. It should be noted that the lexical overlap is recognized by the creators of the corpus (Dolan and Brockett 2005) which indicate a .70 measure of overlap (of an unspecified form). The T-F split in both training and testing is quite similar though (67-33%). Second, the annotations by humans were made on slightly modified sentences which are different from the original sentences publicly released. For instance, humans were asked to ignore all numbers and simply replace them with a generic token, e.g. MONEY for monetary values, and make judgments accordingly. This discrepancy between what humans used and what systems take as input complicates the task as some decisions are counterintuitive which means someone trying to define a set of meaningful features by inspecting a subset of examples may be puzzled by some of the expert decisions. For instance, the pair below was judged as a paraphrase although the percentages as well as the indices (Standard & Poor versus Nasdaq) are quite different.

The broader Standard & Poor's 500 Index .SPX gained 3 points, or 0.39 percent, at 924. The technology-laced Nasdaq Composite Index $< .IXIC >$ *rose 6 points, or 0.41 percent, to 1,498.*

Nevertheless, the MSR corpus is the largest available and most widely used.

We report results using four performance metrics: accuracy (percentage of instances correctly predicted out of all instances), precision (percentage of predicted paraphrases that are indeed true paraphrases), recall (percentage of true paraphrases that were predicted as such), and f-measure (harmonic mean of precision and recall).

Before generating words or sequences of words, several preprocessing steps were applied. Because there is a large space of preprocessing possibilities, we present results with different combinations of preprocessing steps. As results show, the preprocessing steps can lead to significant variations of the overall performance of the proposed approach. The first step in preprocessing is tokenization. We used the tokenizer from SharpNLP, which is the CSharp version of the OpenNLP framework.

Next phase in preprocessing is filtering input tokens. First filtering question is about keeping punctuation or not. Both choices have advantages and disadvantages. Imagine defining a bigram kernel. If we ignore punctuation then bigrams that contain a comma or the end of sentence mark will be ignored. This is a form of generalization resulting in a smaller number of features/dimensions of the space. One might argue that bigrams which include punctuation are important.

Table 1: Performance results on unigram and bigram Kernels (train and test)

Kernel Class	Preprocessing Variants		Performance on Train			Performance on Test		
	Punctuation	Stemming	Acc.	Prec.	Recall	Acc.	Prec.	Recall
Unigram Kernels								
Diss	included	no	87.41	85.13	98.58	71.30	72.48	91.63
	included	yes	85.92	83.72	98.26	71.30	72.42	91.80
DissW	included	no	86.87	84.57	98.55	70.96	72.06	91.98
	included	yes	85.57	83.28	98.40	71.25	72.19	92.33
DissLex	removed	yes	87.56	85.79	97.78	73.39	74.75	90.58
	included	yes	87.24	85.43	97.78	73.51	74.68	91.02
DissWLex	removed	yes	87.27	85.57	97.60	73.62	74.64	91.37
	included	yes	86.80	85.01	97.68	73.10	74.10	91.54
Bigram Kernels								
Diss	removed	no	96.39	95.06	99.85	69.80	69.39	97.65
	removed	yes	96.32	94.96	99.85	69.80	69.47	97.38
DissW	removed	no	96.34	94.99	99.85	69.68	69.33	97.56
	removed	yes	96.32	94.96	99.85	69.91	69.53	97.47
DissLex	included	no	96.57	95.35	99.78	71.07	71.72	93.29
	included	yes	96.42	95.18	99.75	71.30	71.88	93.37
DissWLex	removed	no	96.52	95.22	99.85	70.20	71.06	93.11
	included	yes	96.20	94.89	99.75	71.13	71.79	93.20
Lexical Baseline only								
Lex	included	no	71.96	74.12	89.87	72.64	74.12	90.41
	removed	no	72.35	75.09	88.38	72.70	74.89	88.67
	removed	yes	72.74	75.32	88.70	73.51	75.41	89.28
	included	yes	72.74	75.45	88.41	73.68	75.69	89.01

As an example, let us look at the following two sentences which are considered non-paraphrases in the MSR Corpus: A) *The daily Hurriyet said the raid aimed to foil a Turkish plot to kill an unnamed senior Iraqi official in Kirkuk.* and B) *The daily Hurriyet said the raid aimed to foil a Turkish plot to kill an unnamed senior Iraqi Kurdish official in Kirkuk, but Gul has denied any Turkish plot.* We notice that the second sentence has a comma, which in combination with the token *but* is important to decide about paraphrasing. If we remove the comma, then the resulting bigram *Kirkuk-but* will be less important and less likely to appear in other instances of the corpus than the initial bigram *comma-but*.

Another filtering choice is regarding the elimination of *stopwords*. Stopwords are highly frequent words that occur in most of the documents/instances in a collection (e.g. *the, in*). In some tasks, such as Information Retrieval, they are not important and therefore dropped. For our task, the experiments showed that stop words are actually important and should not be ignored. The third filtering choice is about stemming or not the words. Stemming is another technique to reduce the number of dimensions or features when dealing with texts. However, stemming results in loss of morphological information and therefore of valuable hints that could help us distinguish between paraphrases and non-paraphrases. SVMs classifiers are well known for their ability to deal with large number of features which means either choice is feasible in our case. For example, if we have the following two short text fragments: *Children play.* and *Child plays.*, the difference between *child* and *children* is eliminated when stemming, i.e. reducing words to their base form. In this work, we experimented with all 8 combinations of the above preprocessing steps. We report only the best or more interesting results due to space reasons. In particular, in all the shown results we retained stop words, because removing them led consistently to worse results.

A Simple, yet Effective, Informed Baseline

During our experiments with various configurations of our approach, we found out that a simple lexical feature is very effective to detect paraphrases on the test corpus. For the MSR corpus, this was noticed before by Zhang and Patrick (2005) which use a simple four-feature set in combination with a decision tree learning algorithm. Ours is even simpler and provides better results.

This simple lexical baseline relies on counting the number of common word types between sentences of an instance then normalizing this count by the average length of the two sentences, which is the arithmetic mean of their lengths. The resulted normalized score is then used as the only feature input for an SVM classifier with a linear kernel. This feature basically measures how lexically similar two sentences are. Given its surprising performance, we decided to use this feature in combination with our kernel method. In the Results section we refer to this baseline feature as *Lex*, and when combined with the other kernels we append this label to the names of the other methods.

Table 2: Performance on Kernels enhanced with Lexical baseline (10-fold)

Method	Preprocessing Variants		Unigrams			Bigrams		
	Punctuation	Stemming	Acc.	Prec.	Recall	Acc.	Prec.	Recall
DissLex	included	no	73.45	75.44	89.75	71.78	72.77	92.75
	removed	yes	74.14	76.08	89.78	71.42	72.34	93.05
	included	yes	73.76	75.68	89.88	71.61	72.76	92.33
DissWLex	included	no	73.40	75.29	89.96	71.69	72.72	92.67
	removed	yes	74.14	76.00	89.96	71.38	72.32	93.02
	included	yes	73.78	75.58	90.14	71.56	72.69	92.41

Results

After mapping instances into vectors, the vectors are mapped onto a format accepted by the SVM-Light Library (Joachims 1999), an implementation of SVM classifier induction algorithm, that we used to asses our kernel. Although working in highly-dimensional spaces, due to the sparsity of the vectors and efficiency of computing the dot-product between any two vectors, i.e. the kernel function, the running and testing time are relatively small, in the order of seconds for each reported experiment on a medium class laptop. For all the experiments, the overall running time is about one hour when 10-fold cross-validation is used.

During preliminary experiments, we tested with some simple similarity based kernels, which look at n-grams that are common between the paired sentences, and we also tried combinations of similarity and dissimilarity kernels. The results however showed weaker performance for these kernels, when tested on the MSR Paraphrase corpus. Dissimilarity kernels were the only ones that performed well, and the performance on the test data was significantly improved when the lexical feature was added. Apparently this feature seems to make for a good placeholder of a more complex similarity based kernel.

Table 1 presents results for unigram and bigram based kernels, and also for the simple lexical baseline discussed earlier. A complete evaluation was done for all preprocessing variants, and best results are reported for some of these variants. The tables report accuracy, precision, and recall for the positive (paraphrase) instances on both training and testing data sets. Interestingly, the simple baseline provides results comparative with the best results obtained with the dissimilarity kernel on test data. In the tables, *Diss* refers to the kernel method with binary weights while *DissW* refers to the same method with raw frequency weights. The tables only report results on preprocessing variants which gave best performance on the test data. For unigram kernels, we found out that without the lexical feature, best performance is achieved when punctuation is included, while with lexical feature, best performance is given when stemming is used.

For bigram kernels, the number of dimensions is much larger than for unigram kernels. As a result, the learning capacity for bigram classifiers is greater. Therefore, as noticed, these classifier perform very well on the training data, and not so good on the testing data. One might say that this is a classic case of over-training the classifier. Using a 10-fold cross evaluation on all data, might prove or disprove this assumption. Tables 2 presents results using 10-fold cross-validation for both unigram and bigram based kernels when enhanced with the lexical baseline feature (LEX). To check how much do kernels actually contribute to the performance of the classifiers, it is necessary to check whether the baseline provides similar results with the kernel-based methods even if 10-fold cross-validation is used for evaluation (see Table 3). The results in this case are significantly better when unigram-based kernel methods are used.

Table 3: Performance of Lexical Baseline only (10-fold)

Punctuation	Stemming	Acc.	Prec.	Recall
with	no	72.14	74.63	88.76
removed	no	72.44	74.93	88.73
removed	yes	72.81	75.53	88.17
with	yes	73.00	75.57	88.45

We compare our method in table 4 with results reported by others on their single model approaches. Best results are given by the (Wan et al. 2006) method which was replicated in (Das and Smith 2009). The results initially reported in (Wan et al. 2006) were on incomplete data since several hundred instances from both training and testing had to be removed due to some technical problems encountered on the syntactic parser that was used. We can observe that our method provides competitive results in terms of accuracy and best precision even when the more reliable 10-fold cross-validation method is used for evaluation (x10 in the table). Increasing precision for paraphrase identification on the MSR corpus seems to be more challenging than obtaining high recall. Extremely high recall, in the upper 90s, can be easily obtained with very simple lexical overlap methods enhanced with lexical semantics (see (Mihalcea et al. 2006) and (Wan et al. 2006)).

Conclusions

We presented in this paper a novel approach to the task of paraphrase identification. The approach is based on vectors representing dissimilarities between sentences in an instance. A kernel function was defined which was then used in conjunction with Support Vector Machines to induce a classifier in a highly-dimensional space. Results on the challenging Microsoft Research Paraphrase corpus are competitive in terms of accuracy and precision. One advantage of

Table 4: Comparing results with related work

	Acc.	Prec.	Recall
Corley & Mihalcea, 2005	71.50	72.30	92.50
Zhang & Patrick, 2005	71.90	74.30	88.20
Zhang & Patrick (baseline)	72.30	78.80	79.80
Mihalcea et al., 2006	70.30	69.60	97.70
Qui et al., 2006	72.00	72.50	93.40
Lintean & Rus, 2009	72.06	74.04	89.28
Fernando & Stevenson, 2008	74.10	75.20	91.30
Das & Smith, 2009	73.86	79.57	86.05
Wan et al. (repl. in Das&Smith)	75.42	76.88	90.14
OUR BASELINE	73.68	75.69	89.01
OUR METHOD (on test)	73.62	74.64	91.37
OUR METHOD (on 10-fold)	74.14	76.08	89.78

our method is its robustness and lack of external resources, such as lexical semantics from WordNet, when compared to state-of-the-art approaches. Yet, our method is competitive and most precise. For future work, we plan to develop more complex kernels, based on syntax and word-to-word semantic similarity measures, and experiment with other weighting schemes such as entropy or inverse document frequency (Dumais 1991).

Acknowledgment

This research was supported in part by the Institute for Education Sciences (R305A100875) and National Science Foundation (0938239). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

References

Collins, M. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of ACL, Santa Cruz.*

Corley, C. and Mihalcea, R. 2005. Measuring the semantic similarity of texts. In *Proc. of ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment.*

Dagan, I., Glickman, O., and Magnini, B. 2005. The Pascal Recognising Textual Entailment Challenge. In *Proceedings of the Recognizing Textual Entaiment Challenge Workshop.*

Das, D., and Smith, N.A. 2009. Paraphrase Identification as Probabilistic Quasi-Synchronous Recognition *In Proc. of the Joint Conf. of ACL and NLP*, Singapore, August 2009.

Dolan, W.B., and Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP.*

Dumais, S. 1991. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23,229-236.

Fernando, S., and Stevenson, M. 2008. A Semantic Similarity Approach to Paraphrase Detection. In *Proceedings of the Computational Linguistics UK (CLUK 2008).*

Graesser, A.C., Olney, A., Haynes, B.C., and Chipman, P. 2005 AutoTutor: A cognitive system that simulates a tutor that facilitates learning through mixed-initiative dialogue. *Cognitive Systems: Human Cognitive Models in Systems Design* Erlbaum, Mahwah, NJ.

Ibrahim, A.; Katz, B.; and Lin, J. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the 2nd Int. Workshop on Paraphrasing (ACL2003).*

Joachims, T. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning* MIT Press.

Kate, R.J. 2008. A Dependency-based Word Subsequence Kernel. In *Proceedings of EMNLP, 2008.*

Lintean, M., and Rus, V. 2009. Paraphrase Identification Using Weighted Dependencies and Word Semantics. *Proceedings of the FLAIRS-22.* Sanibel Island, FL.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. 2002. Text Classification using String Kernels. *Journal of Machine Learning Research,* 2:419–444.

Mihalcea, R., Corley, C., and Strapparava, C. 2006. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In *Proceedings of AAAI-2006.* Boston, July.

Pedersen, T., Patwardhan, S., and Michelizzi, J. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proceedings of AAAI-04.* San Jose, CA.

Rus, V., Nan, X., Shiva, S.G., and Chen, Y. 2009. Clustering of Defect Reports Using Graph Partitioning Algorithms. In *Proceedings of SEKE'2009.* Boston, MA, July, p. 442–445.

Rus, V., Lintean, M., Graesser, A.C., McNamara, D.S. 2009. Assessing Student Paraphrases Using Lexical Semantics and Word Weighting In *Proceedings of AIED.* Brighton, UK.

Qiu, L., Kan, M. Y., and Chua, T. S. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of EMNLP.*

Shawe-Taylor, J., and Cristianini, N. 2000. *Support Vector Machines and other kernel-based learning methods.* Cambridge University Press.

Wan, S., Dras, M., Dale, M., and Paris, C. 2006. Using dependency-based features to take the para-farce out of paraphrase *In Proceedings of ALTW.*

Weeds, J., Weir, D., and Keller, B. 2005. The distributional similarity of sub-parses. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment.* Ann Arbor, MI, June, ACL, p. 7–12.

Wu, D. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment.*

Zanzotto, F. M., Pannacchiotti, M., and Moschitti, A. 2009. A machine learning approach to textual entailnment recognition. *Natural Language Engineering*, 15(4):551–582.

Zhang, Y., and Patrick, J. 2005. Paraphrase identification by text canonicalization. *In Proceedings of ALTW*,160–166.