TakeLab: Systems for Measuring Semantic Text Similarity

Frane Šarić, Goran Glavaš, Mladen Karan,
Jan Šnajder, and Bojana Dalbelo Bašić
University of Zagreb
Faculty of Electrical Engineering and Computing
{frane.saric, goran.glavas, mladen.karan, jan.snajder, bojana.dalbelo}@fer.hr

Abstract

This paper describes the two systems for determining the semantic similarity of short texts submitted to the SemEval 2012 Task 6. Most of the research on semantic similarity of textual content focuses on large documents. However, a fair amount of information is condensed into short text snippets such as social media posts, image captions, and scientific abstracts. We predict the human ratings of sentence similarity using a support vector regression model with multiple features measuring word-overlap similarity and syntax similarity. Out of 89 systems submitted, our two systems ranked in the top 5, for the three overall evaluation metrics used (*overall Pearson* – 2nd and 3rd, *normalized Pearson* – 1st and 3rd, *weighted mean* – 2nd and 5th).

1 Introduction

Natural language processing tasks such as text classification (Sebastiani, 2002), text summarization (Lin and Hovy, 2003; Aliguliyev, 2009), information retrieval (Park et al., 2005), and word sense disambiguation (Schütze, 1998) rely on a measure of semantic similarity of textual documents. Research predominantly focused either on the document similarity (Salton et al., 1975; Maguitman et al., 2005) or the word similarity (Budanitsky and Hirst, 2006; Agirre et al., 2009). Evaluating the similarity of short texts such as sentences or paragraphs (Islam and Inkpen, 2008; Mihalcea et al., 2006; Oliva et al., 2011) received less attention from the research community. The task of recognizing paraphrases (Michel et al., 2011; Socher et al., 2011; Wan et al., 2006) is sufficiently similar to reuse some of the techniques.

This paper presents the two systems for automated measuring of semantic similarity of short texts which we submitted to the SemEval-2012 Semantic Text Similarity Task (Agirre et al., 2012). We propose several sentence similarity measures built upon knowledge-based and corpus-based similarity of individual words as well as similarity of dependency parses. Our two systems, *simple* and *syntax*, use supervised machine learning, more specifically the support vector regression (SVR), to combine a large amount of features computed from pairs of sentences. The two systems differ in the set of features they employ.

Our systems placed in the top 5 (out of 89 submitted systems) for all three aggregate correlation measures: 2nd (*syntax*) and 3rd (*simple*) for overall Pearson, 1st (*simple*) and 3rd (*syntax*) for normalized Pearson, and 2nd (*simple*) and 5th (*syntax*) for weighted mean.

The rest of the paper is structured as follows. In Section 2 we describe both knowledge-based and corpus-based word similarity measures. In Section 3 we describe in detail the features used by our systems. In Section 4 we report the experimental results cross-validated on the development set as well as the official results on all test sets. Conclusions and ideas for future work are given in Section 5.

2 Word Similarity Measures

Approaches to determining semantic similarity of sentences commonly use measures of semantic sim-

*First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 441–448,
Montréal, Canada, June 7-8, 2012.

ilarity between individual words. Our systems use the knowledge-based and the corpus-based (i.e., distributional lexical semantics) approaches, both of which are commonly used to measure the semantic similarity of words.

2.1 Knowledge-based Word Similarity

Knowledge-based word similarity approaches rely on a semantic network of words, such as WordNet. Given two words, their similarity can be estimated by considering their relative positions within the knowledge base hierarchy.

All of our knowledge-based word similarity measures are based on WordNet. Some measures use the concept of a *lowest common subsumer (LCS)* of concepts c_1 and c_2, which represents the lowest node in the WordNet hierarchy that is a hypernym of both c_1 and c_2. We use the NLTK library (Bird, 2006) to compute the PathLen similarity (Leacock and Chodorow, 1998) and Lin similarity (Lin, 1998) measures. A single word often denotes several concepts, depending on its context. In order to compute the similarity score for a pair of words, we take the maximum similarity score over all possible pairs of concepts (i.e., WordNet synsets).

2.2 Corpus-based Word Similarity

Distributional lexical semantics models determine the meaning of a word through the set of all contexts in which the word appears. Consequently, we can model the meaning of a word using its distribution over all contexts. In the distributional model, deriving the semantic similarity between two words corresponds to comparing these distributions. While many different models of distributional semantics exist, we employ latent semantic analysis (LSA) (Turney and Pantel, 2010) over a large corpus to estimate the distributions.

For each word w_i, we compute a vector x_i using the truncated singular value decomposition (SVD) of a tf-idf weighted term-document matrix. The cosine similarity of vectors x_i and x_j estimates the similarity of the corresponding words w_i and w_j.

Two different word-vector mappings were computed by processing the New York Times Annotated Corpus (NYT) (Sandhaus, 2008) and Wikipedia. Aside from lowercasing the documents and removing punctuation, we perform no further preprocessing (e.g., no stopwords removal or stemming). Upon removing the words not occurring in at least two documents, we compute the tf-idf. The word vectors extracted from NYT corpus and Wikipedia have a dimension of 200 and 500, respectively.

Table 1: Evaluation of word similarity measures

Measure	ws353	ws353-sim	ws353-rel
PathLen	0.29	0.61	-0.05
Lin	0.33	0.64	-0.01
Dist (NYT)	0.50	0.50	0.51
Dist (Wikipedia)	0.62	0.66	0.55

We compared the measures by computing the Spearman correlation coefficient on the WordSim353[1] data set, as well as its similarity and relatedness subsets described in (Agirre et al., 2009). Table 1 provides the results of the comparison.

3 Semantic Similarity of Sentences

Our systems use supervised regression with SVR as a learning model, where each system exploits different feature sets and SVR hyperparameters.

3.1 Preprocessing

We list all of the preprocessing steps our systems perform. If a preprocessing step is executed by only one of our systems, the system's name is indicated in parentheses.

1. All hyphens and slashes are removed;
2. The angular brackets (< and >) that enclose the tokens are stripped (*simple*);
3. The currency values are simplified, e.g., `$US1234` to `$1234` (*simple*);
4. Words are tokenized using the Penn Treebank compatible tokenizer;
5. The tokens *n't* and *'m* are replaced with *not* and *am*, respectively (*simple*);
6. The two consecutive words in one sentence that appear as a compound in the other sentence are replaced by the said compound. E.g., *cater pillar* in one sentence is replaced with *caterpillar* only if *caterpillar* appears in the other sentence;

[1] `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/wordsim353.html`

7. Words are POS-tagged using Penn Treebank compatible POS-taggers: NLTK (Bird, 2006) for *simple*, and OpenNLP[2] for *syntax*;
8. Stopwords are removed using a list of 36 stopwords (*simple*).

While we acknowledge that some of the preprocessing steps we take may not be common, we did not have the time to determine the influence of each individual preprocessing step on the results to either warrant their removal or justify their presence.

Since, for example, *sub-par*, *sub par* and *subpar* are treated as equal after preprocessing, we believe it makes our systems more robust to inputs containing small orthographic differences.

3.2 Ngram Overlap Features

We use many features previously seen in paraphrase classification (Michel et al., 2011). Several features are based on the unigram, bigram, and trigram overlap. Before computing the overlap scores, we remove punctuation and lowercase the words. We continue with a detailed description of each individual feature.

Ngram Overlap

Let S_1 and S_2 be the sets of consecutive ngrams (e.g., bigrams) in the first and the second sentence, respectively. The ngram overlap is defined as follows:

$$ngo(S_1, S_2) = 2 \cdot \left(\frac{|S_1|}{|S_1 \cap S_2|} + \frac{|S_2|}{|S_1 \cap S_2|} \right)^{-1} \quad (1)$$

The ngram overlap is the harmonic mean of the degree to which the second sentence covers the first and the degree to which the first sentence covers the second. The overlap, defined by (1), is computed for unigrams, bigrams, and trigrams.

Additionally we observe the *content ngram overlap* – the overlap of unigrams, bigrams, and trigrams exclusively on the content words. The content words are nouns, verbs, adjectives, and adverbs, i.e., the lemmas having one of the following part-of-speech tags: JJ, JJR, JJS, NN, NNP, NNS, NNPS, RB, RBR, RBS, VB, VBD, VBG, VBN, VBP, and VBZ. Intuitively, the function words (prepositions, conjunctions, articles) carry less semantics than content words and thus removing them might eliminate the noise and provide a more accurate estimate of semantic similarity.

In addition to the overlap of consecutive ngrams, we also compute the skip bigram and trigram overlap. Skip-ngrams are ngrams that allow arbitrary gaps, i.e., ngram words need not be consecutive in the original sentence. By redefining S_1 and S_2 to represent the sets of skip ngrams, we employ eq. (1) to compute the skip-n gram overlap.

3.3 WordNet-Augmented Word Overlap

One can expect a high unigram overlap between very similar sentences only if exactly the same words (or lemmas) appear in both sentences. To allow for some lexical variation, we use WordNet to assign partial scores to words that are not common to both sentences. We define the WordNet augmented coverage $P_{WN}(\cdot, \cdot)$:

$$P_{WN}(S_1, S_2) = \frac{1}{|S_2|} \sum_{w_1 \in S_1} score(w_1, S_2)$$

$$score(w, S) = \begin{cases} 1 & \text{if } w \in S \\ \max_{w' \in S} sim(w, w') & \text{otherwise} \end{cases}$$

where $sim(\cdot, \cdot)$ represents the WordNet path length similarity. The *WordNet-augmented word overlap* feature is defined as a harmonic mean of $P_{WN}(S_1, S_2)$ and $P_{WN}(S_2, S_1)$.

Weighted Word Overlap

When measuring sentence similarities we give more importance to words bearing more content, by using the information content

$$ic(w) = \ln \frac{\sum_{w' \in C} freq(w')}{freq(w)}$$

where C is the set of words in the corpus and $freq(w)$ is the frequency of the word w in the corpus. We use the Google Books Ngrams (Michel et al., 2011) to obtain word frequencies because of its excellent word coverage for English. Let S_1 and S_2 be the sets of words occurring in the first and second sentence, respectively. The weighted word coverage of the second sentence by the first sentence is

[2] http://opennlp.apache.org/

given by:

$$wwc(S_1, S_2) = \frac{\sum_{w \in S_1 \cap S_2} ic(w)}{\sum_{w' \in S_2} ic(w')}$$

The *weighted word overlap* between two sentences is calculated as the harmonic mean of the $wwc(S_1, S_2)$ and $wwc(S_2, S_1)$.

This measure proved to be very useful, but it could be improved even further. Misspelled frequent words are more frequent than some correctly spelled but rarely used words. Hence dealing with misspelled words would remove the inappropriate heavy penalty for a mismatch between correctly and incorrectly spelled words.

Greedy Lemma Aligning Overlap

This measure computes the similarity between sentences using the semantic alignment of lemmas. First we compute the word similarity between all pairs of lemmas from the first and the second sentence, using either the knowledge-based or the corpus-based semantic similarity. We then greedily search for a pair of most similar lemmas; once the lemmas are paired, they are not considered for further matching. Previous research by Lavie and Denkowski (2009) proposed a similar alignment strategy for machine translation evaluation. After aligning the sentences, the similarity of each lemma pair is weighted by the larger information content of the two lemmas:

$$sim(l_1, l_2) = \max(ic(l_1), ic(l_2)) \cdot ssim(l_1, l_2) \quad (2)$$

where $ssim(l_1, l_2)$ is the semantic similarity between lemmas l_1 and l_2.

The overall similarity between two sentences is defined as the sum of similarities of paired lemmas normalized by the length of the longer sentence:

$$glao(S_1, S_2) = \frac{\sum_{(l_1,l_2) \in P} sim(l_1, l_2)}{\max(length(S_1), length(S_2))}$$

where P is the set of lemma pairs obtained by greedy alignment. We take advantage of greedy align overlap in two features: one computes $glao(\cdot, \cdot)$ by using the Lin similarity for $ssim(\cdot, \cdot)$ in (2), while the other feature uses the distributional (LSA) similarity to calculate $ssim(\cdot, \cdot)$.

Vector Space Sentence Similarity

This measure is motivated by the idea of compositionality of distributional vectors (Mitchell and Lapata, 2008). We represent each sentence as a single distributional vector $u(\cdot)$ by summing the distributional (i.e., LSA) vector of each word w in the sentence S: $u(S) = \sum_{w \in S} \mathbf{x}_w$, where $\mathbf{x}_w$ is the vector representation of the word w. Another similar representation $u_W(\cdot)$ uses the information content $ic(w)$ to weigh the LSA vector of each word before summation: $u_W(S) = \sum_{w \in S} ic(w)\mathbf{x}_w$. The *simple* system uses $|\cos(u(S_1), u(S_2))|$ and $|\cos(u_W(S_1), u_W(S_2))|$ for the *vector space sentence similarity* features.

3.4 Syntactic Features

We use dependency parsing to identify the lemmas with the corresponding syntactic roles in the two sentences. We also compute the overlap of the dependency relations of the two sentences.

Syntactic Roles Similarity

The similarity of the words or phrases having the same syntactic roles in the two sentences may be indicative of their overall semantic similarity (Oliva et al., 2011). For example, two sentences with very different main predicates (e.g., *play* and *eat*) probably have a significant semantic difference.

Using Lin similarity $ssim(\cdot, \cdot)$, we obtain the similarity between the matching lemmas in a sentence pair for each syntactic role. Additionally, for each role we compute the similarity of the chunks that lemmas belong to:

$$chunksim(C_1, C_2) = \sum_{l_1 \in C_1} \sum_{l_2 \in C_2} ssim(l_1, l_2)$$

where C_1 and C_2 are the sets of chunks of the first and second sentence, respectively. The final similarity score of two chunks is the harmonic mean of $chunksim(C_1, C_2)/|C_1|$ and $chunksim(C_1, C_2)/|C_2|$.

Syntactic roles that we consider are predicates (p), subjects (s), direct (d), and indirect (i) (i.e., prepositional) objects, where we use (o) to mean either (d) or (i). The Stanford dependency parser (De Marneffe et al., 2006) produces the dependency parse of the sentence. We infer (p), (s), and (d) from the syntactic dependencies of type *nsubj* (nominal subject),

nsubjpass (nominal subject passive), and *dobj* (direct object). By combining the *prep* and *pobj* dependencies (De Marneffe and Manning, 2008), we identify (i). Since the (d) in one sentence often semantically corresponds to (i) in the other sentence, we pair all (o) of one sentence with all (o) of the other sentence and define object similarity between the two sentences as the maximum similarity among all (o) pairs. Because the syntactic role might be absent from both sentences (e.g., the object in sentences "John sings" and "John is thinking"), we introduce additional binary features indicating if the comparison for the syntactic role in question exists.

Many sentences (especially longer ones) have two or more (p). In such cases it is necessary to align the corresponding predicate groups (i.e., the (p) with its corresponding arguments) between the two sentences, while also aggregating the (p), (s), and (o) similarities of all aligned (p) pairs. The similarity of two predicate groups is defined as the sum of (p), (s), and (o) similarities. In each iteration, the greedy algorithm pairs all predicate groups of the first sentence with all predicate groups of the second sentence and searches for a pair with the maximum similarity. Once the predicate groups of two sentences have been aligned, we compute the (p) similarity as a weighted sum of (p) similarities for each predicate pair group. The weight of each predicate group pair equals the larger information content of two predicates. The (s) and (o) similarities are computed in the same manner.

Syntactic Dependencies Overlap

Similar to the ngram overlap features, we measure the overlap between sentences based on matching dependency relations. A similar measure has been proposed in (Wan et al., 2006). Two syntactic dependencies are considered equal if they have the same dependency type, governing lemma, and dependent lemma. Let S_1 and S_2 be the set of all dependency relations in the first and the second sentence, respectively. *Dependency overlap* is the harmonic mean between $|S_1 \cap S_2|/|S_1|$ and $|S_1 \cap S_2|/|S_2|$. *Content dependency overlap* computes the overlap in the same way, but considers only dependency relations between content lemmas.

Similarly to weighted word overlap, we compute the weighted dependency relations overlap. The weighted coverage of the second sentence dependencies with the first sentence dependencies is given by:

$$wdrc(S_1, S_2) = \frac{\sum_{r \in S_1 \cap S_2} \max(ic(g(r)), ic(d(r)))}{\sum_{r \in S_2} \max(ic(g(r)), ic(d(r)))}$$

where $g(r)$ is the governing word of the dependency relation r, $d(r)$ is the dependent word of the dependency relation r, and $ic(l)$ is the information content of the lemma l. Finally, the *weighted dependency relations overlap* is the harmonic mean between $wdrc(S_1, S_2)$ and $wdrc(S_2, S_1)$.

3.5 Other Features

Although we primarily focused on developing the ngram overlap and syntax-based features, some other features significantly improve the performance of our systems.

Normalized Differences

Our systems take advantage of the following features that measure normalized differences in a pair of sentences: (A) sentence length, (B) the noun chunk, verb chunk, and predicate counts, and (C) the aggregate word information content (see *Normalized differences* in Table 2).

Numbers Overlap

The annotators gave low similarity scores to many sentence pairs that contained different sets of numbers, even though their sentence structure was very similar. Socher et al. (2011) improved the performance of their paraphrase classifier by adding the following features that compare the sets of numbers N_1 and N_2 in two sentences: $N_1 = N_2$, $N_1 \cap N_2 \neq \emptyset$, and $N_1 \subseteq N_2 \vee N_2 \subseteq N_1$. We replace the first two features with $\log(1 + |N_1| + |N_2|)$ and $2 \cdot |N_1 \cap N_2|/(|N_1| + |N_2|)$. Additionally, the numbers that differ only in the number of decimal places are treated as equal (e.g., 65, 65.2, and 65.234 are treated as equal, whereas 65.24 and 65.25 are not).

Named Entity Features

Shallow NE similarity treats capitalized words as named entities if they are longer than one character. If a token in all caps begins with a period, it is classified as a stock index symbol. The *simple* system

Table 2: The usage of feature sets

Feature set	*simple*	*syntax*
Ngram overlap	+	+
Content-ngram overlap	-	+
Skip-ngram overlap	-	+
WordNet-aug. overlap	+	-
Weighted word overlap	+	+
Greedy align. overlap	-	+
Vector space similarity	+	-
Syntactic roles similarity	-	+
Syntactic dep. overlap	-	+
Normalized differences*	A,C	A,B
Shallow NERC	+	-
Full NERC	-	+
Numbers overlap	+	+

* See Section 3.5

uses the following four features: the overlap of capitalized words, the overlap of stock index symbols, and the two features indicating whether these named entities were found in either of the two sentences.

In addition to the overlap of capitalized words, the *syntax* system uses the OpenNLP named entity recognizer and classifier to compute the overlap of entities for each entity class separately. We recognize the following entity classes: persons, organizations, locations, dates, and rudimentary temporal expressions. The absence of an entity class from both sentences is indicated by a separate binary feature (one feature for each class).

Feature Usage in TakeLab Systems

Some of the features presented in the previous sections were used by both of our systems (*simple* and *syntax*), while others were used by only one of the systems. Table 2 indicates the feature sets used for the two submitted systems.

4 Results

4.1 Model Training

For each of the provided training sets we trained a separate Support Vector Regression (SVR) model using LIBSVM (Chang and Lin, 2011). To obtain the optimal SVR parameters C, g, and p, our systems employ a grid search with nested cross-validation. Table 3 presents the cross-validated performance (in terms of Pearson correlation) on the training sets. The models tested on the SMTnews test set were trained on the SMTeuroparl train set. For the OnWn test set, the *syntax* model was trained on the MSRpar set, while the *simple* system's model was trained on the union of all train sets. The final predictions were trimmed to a 0–5 range.

Table 3: Cross-validated results on train sets

	MSRvid	MSRpar	SMTeuroparl
simple	0.8794	0.7566	0.7802
syntax	0.8698	0.7144	0.7308

Our development results indicate that the *weighted word overlap*, *WordNet-augmented word overlap*, the *greedy lemma alignment overlap*, and the *vector space sentence similarity* individually obtain high correlations regardless of the development set in use. Other features proved to be useful on individual development sets (e.g., *syntax roles similarity* on MSRvid and *numbers overlap* on MSRpar). More research remains to be done in thorough feature analysis and systematic feature selection.

4.2 Test Set Results

The organizers provided five different test sets to evaluate the performance of the submitted systems. Table 4 illustrates the performance of our systems on individual test sets, accompanied by their rank. Our systems outperformed most other systems on MSRvid, MSRpar, and OnWN sets (Agirre et al., 2012). However, they performed poorly on the SMTeuroparl and SMTnews sets. While the correlation scores on the MSRvid and MSRpar test sets correspond to those obtained using cross-validation on the corresponding train sets, the performance on the SMT test sets is drastically lower than the cross-validated performance on the corresponding train set. The sentences in the SMT training set are significantly longer (30.4 tokens on average) than the sentences in both SMT test sets (12.3 for SMTeuroparl and 13.5 for SMTnews). Also there are several repeated pairs of extremely short and identical sentences (e.g., "Tunisia" – "Tunisia" appears 17 times

Table 4: Results on individual test sets

	simple	*syntax*
MSRvid	0.8803 (1)	0.8620 (8)
MSRpar	0.7343 (1)	0.6985 (2)
SMTeuroparl	0.4771 (26)	0.3612 (63)
SMTnews	0.3989 (46)	0.4683 (18)
OnWN	0.6797 (9)	0.7049 (6)

Table 5: Aggregate performance on the test sets

	All	ALLnrm	Mean
simple	0.8133 (3)	0.8635 (1)	0.6753 (2)
syntax	0.8138 (2)	0.8569 (3)	0.6601 (5)

in the SMTeuroparl test set). The above measurements indicate that the SMTeuroparl training set was not representative of the SMTeuroparl test set for our choice of features.

Table 5 outlines the aggregate performance of our systems according to the three aggregate evaluation measures proposed for the task (Agirre et al., 2012). Both systems performed very favourably compared to the other systems, achieving very high rankings regardless of the aggregate evaluation measure.

The implementation of *simple* system is available at `http://takelab.fer.hr/sts`.

5 Conclusion and Future Work

In this paper we described our submission to the SemEval-2012 Semantic Textual Similarity Task. We have identified some high performing features for measuring semantic text similarity. Although both of the submitted systems performed very well on all but the two SMT test sets, there is still room for improvement. The feature selection was ad-hoc and more systematic feature selection is required (e.g., wrapper feature selection). Introducing additional features for deeper understanding (e.g., semantic role labelling) might also improve performance on this task.

Acknowledgments

This work was supported by the Ministry of Science, Education and Sports, Republic of Croatia under the Grant 036-1300646-1986. We would like to thank the organizers for the tremendous effort they put into formulating this challenging task.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*. ACL.

Ramiz M. Aliguliyev. 2009. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Expert Systems with Applications*, 36(4):7764–7772.

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL '06, pages 69–72. Association for Computational Linguistics.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Marie-Catherine De Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10.

Alon Lavie and Michael Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine translation*, 23(2):105–115.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th international conference on Machine Learning*, volume 1, pages 296–304. San Francisco.

Ana G. Maguitman, Filippo Menczer, Heather Roinestad, and Alessandro Vespignani. 2005. Algorithmic detection of semantic similarity. In *Proceedings of the 14th international conference on World Wide Web*, pages 107–116. ACM.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva P. Aiden, Adrian Veres, Matthew K. Gray, et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the national conference on artificial intelligence*, volume 21, page 775. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *Proceedings of ACL-08: HLT*, pages 236–244.

Jesús Oliva, Jóse Ignacio Serrano, María Dolores Del Castillo, and Ángel Iglesias. 2011. SyMSS: A syntax-based measure for short-text semantic similarity. *Data & Knowledge Engineering*.

Eui-Kyu Park, Dong-Yul Ra, and Myung-Gil Jang. 2005. Techniques for improving web retrieval effectiveness. *Information processing & management*, 41(5):1207–1223.

Gerard Salton, Andrew Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics*, 24(1):97–123.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in Neural Information Processing Systems*, 24.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.