NORMAS at SemEval-2016 Task 1: SEMSIM: A Multi-Feature Approach to Semantic Text Similarity

Adebayo Kolawole John and **Luigi Di Caro** and **Guido Boella**
University of Torino
Corso Svizzera 185
Torino, 10149, Italy
kolawolejohn.adebayo@unibo.it, (dicaro, guido)@di.unito.it

Abstract

This paper presents the submission of our team (NORMAS) to the SemEval 2016 semantic textual similarity (STS) shared task. We submitted three system runs, each using a set of 36 features extracted from the training set. The runs explore the use of the following three machine learning algorithms: Support Vector Regression, Elastic Net and Random Forest. Each run was trained using sentence pairs from the STS 2012 training data. Features extracted include lexical, syntactic and semantic features. This paper describes the features we designed for assessing the semantic similarity between sentence pairs, the models we build using these features and the performance obtained by the resulting systems on the 2016 evaluation data.

1 Introduction

Computationally assessing the semantic similarity of natural language data has gained the attention of researchers in the field of computational linguistics. Machines that are able to quantify the semantics of natural language have numerous important applications including: Document Similarity (Elsayed et al., 2008; Huang, 2008), Word Similarity (Resnik, 1995; Dagan, 2000; Bollegala et al., 2007; Pedersen et al., 2004), Text Summarization (Barzilay and Elhadad, 1999; Gong and Liu, 2001), Informtion Retrieval (Salton and Buckley, 1988; Manning et al., 2008), Plagiarism detection (Si et al., 1997), Paraphrase detection (Fernando and Stevenson, 2008) and especially Machine Translation (Brown et al., 1990).

The semantic textual similarity (STS) (Agirre et al., 2012) task measures the level of semantic equivalence between two approximately sentence sized snippets of texts on a graded scale from 0 (unrelated) to 5 (completely equivalent). This paper describes our participation in the SemEval 2016 STS shared task (Agirre et al., 2016). Our system explores 36 lexical and semantic features (e.g., string matching, WordNet similarity, word overlap) in combination with three very distinct learning algorithms (Support Vector Regression, Elastic Net and Random forest).

The remainder of this paper is organized as follows: *Section 2* describes our feature set in detail and our approach to feature selection. *Section 3* describes our machine learning models with section 4 presenting our results on the shared task evaluation data.

2 Feature Generation

We used a mixture of lexical, syntactic and semantic features extracted from text. Below, we describe each of the features provided to our model.

2.1 Lexical Features

String matching: String matching techniques compare words in two texts character by character and can be used to approximately capture morphological differences in the terms. Bär et al. (2012), one of the best performing systems in 2012, used variants of string matching algorithms. Our system made use of features computed using the following string matching methods:

1. ***Longest common Substring***: This obtains the longest sequences of words appearing in both

Proceedings of SemEval-2016, pages 718–725,
San Diego, California, June 16-17, 2016. ©2016 Association for Computational Linguistics

sentences (Gusfield, 1997).

2. ***Levenshtein Distance***: This measures the number of basic edit-operations (insertions, deletions, and substitutions) required to change one text into the other.

3. ***Jaccard Similarity***: This measures the number of words shared by two sentences in ratio with the total number of words in the sentences i.e. given sentences A and B, the ratio is defined as:

$$Jac = \frac{A \cap B}{A \cup B} \tag{1}$$

Word Ordering: We use the union of all tokens in a pair of sentences to build a vocabulary of non-repeating terms. For each sentence, the position mapping of each word in the vocabulary is used to build a vector. To obtain the position mapping, a unique index number is assigned to each vocabulary term. To obtain the word order vector for a sentence, each term in the vocabulary is compared against terms in the sentence. If a vocabulary term is found in the sentence, the index number of that term in the vocabulary is added to the vector. Otherwise, similarity of the vocabulary term and each term in the sentence is calculated using a WordNet based word similarity algorithm. The index number of the sentence term with highest similarity score above a threshold is added. If the first two conditions does not hold, 0 is added to the vector. Consider two sentences S1 and S2,

S1: A panda bear

S2: A baby panda

Then the vocabulary is a list that contains the union of tokens in *S1* and *S2* as shown below:

Vocabulary = A, baby, bear, panda

Vocabulary-Index = A:1, baby:2, bear:3, panda:4

and the sentences are transformed to the vectors below:

S1 = 1,0,3,4

S2 = 1,2,4,4

In the example, the vocabulary term *bear* does not exist in *S2*. However, *bear* is closer to *panda* than all the terms in *S2*. The similarity score between them also exceeds the threshold. The index number of *panda* is thus assigned in place of *bear*. In *S1*, the vocabulary term *baby* is not similar to any term, thus 0 is assigned. The word ordering feature is then computed as the cosine of the vectors after the WordNet based similarity transformation.

Word Overlap: We use the word n-gram overlap features of Šarić et al. (2012). The n-grams overlap is defined as the harmonic mean of the degree of mappings between the first and second sentence and vice versa, requiring an exact string match of n-grams in the two sentences.

$$\text{Ng(A,B)} = \left(2(\frac{\mid A \mid}{A \cap B} + \frac{\mid B \mid}{A \cap B})^{-1}\right) \tag{2}$$

Where *A* and *B* are the set of n-grams in the two sentences. We computed three separate features using equation 2 for each of the following character n-grams: unigram, bigrams and trigrams. We also use weighted word overlap which uses information content (Resnik, 1995).

$$\text{wwo(A,B)} = \frac{\sum_{w \in A \cap B} ic(w)}{\sum_{w' \in B} ic(w')} \tag{3}$$

$$\text{ic(w)} = \left(\ln \frac{\sum_{w' \in C} freq(w')}{freq(w)}\right) \tag{4}$$

Where *C* is the set of words and *freq(w)* is the occurrence count obtained from the Brown corpus. Our weighted word overlap feature is computed as the harmonic mean of the functions *wwo(A,B)* and *wwo(B,A)*.

Entity Overlap: When two sentences have Named Entities (NEs) in common, a semblance of similarity is reflected. We extracted NEs from each sentence using the Stanford NER tagger (Finkel et al., 2005; Manning et al., 2014). The entity overlap feature is obtained as follows:

$$\text{EOV} = \frac{|A \tilde{\cap} B|}{|A \cup B|} \tag{5}$$

Where *A* and *B* represent the set of named entities in the first and second sentences, respectively. The intersection, $\tilde{\cap}$, allows partial matches since the NEs are considered equivalent if either there is an exact match or if one NE is a substring of the other. For example, '*President Obama*' is not the same as '*Barack Obama*', but '*Obama*' is considered a match for either of these.

2.2 Syntactic Features

POS Overlap: We used the Stanford POS-tagger to tag the words in each sentence with their POS categories. We group the words by the following coarse grained POS categories: nouns, verbs, adjectives and adverbs. We then compare the overlap between these classes of part of speech in each sentence, e.g., we compare the nouns in sentence 1 to the nouns in sentence 2 and vice versa across all 4 coarse grained POS categories. The POS overlap features are defined as the Jaccard similarity of the two sentences across just the words in a particular POS category:

$$POV_{pos} = \frac{|A_{pos} \cap B_{pos}|}{|A_{pos} \cup B_{pos}|} \tag{6}$$

Where A_{pos} and B_{pos} are the set of terms in POS class pos of sentences 1 and 2, respectively. This results in 4 unique POS overlap features.

Dependency Parsing: It is apt to assume that shared named entities can point to similarity in sentences. This assumption is more valid if sentences share the same *subject* and *object*. For example, if a named entity that is a *subject* in sentence1 is also the *subject* in sentence2 and vice versa the object. As an example, consider the two sentences below:

S1: Obama is the president of the United States

S2: Obama is the leader of the United States

In the first sentence, *Obama* is the subject (n-subj) of *President* while *State* modifies (nmod) the *President*. Also in the second sentence *Obama* is the subject of *leader* while *State* modifies the word *leader*.

We used the well-known *Stanford Parser* (Manning et al., 2014; De Marneffe et al., 2006) to extract the *subject-verb-object* triples from each sentence. In particular, the Neural Network-based dependency parser (Chen and Manning, 2014) was employed. We compare the *subject* and *object* of both sentences. If any of the objects or subjects in a sentence is a NE, we simply compare with the corresponding one from the other sentence by pure string matching.When the same word takes either the role of *subject* or *object* in the two sentences, we assign a flat score of 0.5. If both the *subject* and *object* in the two sentences correspond to the same NEs as in the example above, we assign a score of 1.0. Otherwise, we assign a score of 0.0.

2.3 Semantic Features

We extracted some semantic features using both information induced from corpus data as well as knowledge from existing semantic resources as described below:

Word Embedding Similarity: Word2Vec[1] (Mikolov et al., 2013b; Mikolov et al., 2013a) is an algorithm for inducing vector space representations of words, commonly known as word embeddings, that capture semantic relatedness and similarity. The method exploits the Distributional Hypothesis (Turney et al., 2010) and works best when trained on large corpora. We used the Gensim[2] implementation of the algorithm with specific parameter fine-tuning.[3] Gensim implements a variant of the algorithm known as *Skip-Gram,* which trains word representations using a model that when given a word will predict what words are likely to occur within a fixed window around it.[4] Once the word embeddings have been trained, the similarity of two words is computed as the cosine of the two embedding vectors.[5] Our word embeddings are trained on a combination of the Wikipedia dump of English language articles and the STS 2016 training data.[6]

For sentence level similarity scores, we compare each word in the first sentence to each word in the second sentence, obtaining a similarity score with the word2vec model. For each pair being compared, if the similarity score is less than < 0.25 then that similarity value is dropped. The final similarity is computed by summing the pair similarity values greater than 0.25 and dividing by the total count of these similarity scores. The aggregation function is

[1] Word2vec is available at https://code.google.com/p/word2vec/

[2] Gensim is a python library for an array of NLP tasks. It is available at https://radimrehurek.com/gensim/

[3] Parameters used: Context Window: 5, Neural Network layer size: 200, Minimum word count: 5.

[4] Skip-gram training generally performs better than an alternative Word2Vec model known as Continuous-Bag-of-Words (CBOW) that uses an alternative objective that tries to predict a word by conditioning on all of the words that surround it within a window.

[5] model.similarity(word1, word2) returns a similarity value from -1 to 1 between word1 and word2. Also model[word1] returns a numpy vector of word1

[6] The wikipedia dump was downloaded on July 30, 2015. It is accessible at https://dumps.wikimedia.org/enwiki/

given below:

$$\text{Sim} = \frac{\sum_{i,j}^{m,n} |S(w_i, w_j > x)|}{tCount} \tag{7}$$

Where *S(w_i, w_j)* is the similarity score for two words, *tCount* is the total number of the set of similarity scores that exceeds the threshold and *Sim* is the aggregating function combining all pairwise similarities.

WordNet similarity: The idea of pairing and aggregating similarity of words from two sentences being compared is not new and has been used both in textual entailment (Agichtein et al., 2008) and semantic textual similarity (Bär et al., 2012). To derive similarity from WordNet, we used both the path length between each word as well as the depth function. Usually, longer path length between two concepts signifies lower similarity. However, as pointed out by Li et al. (2006), this obviates the distance knowledge that can be easily observed from the hierarchical organization of some *semantic nets*. As a solution, the depth function was introduced, with the intuition that words at upper layer of a *semantic nets* contains general semantics and less similarity while those at lower layers are more similar. Thus, similarity should be a function of both the depth and the path length distances between concepts. If f1(h) is a function of the depth and f2(l) is a function of the length, then the similarity between two word is given by:

$$S(w_1, w_2) = f1(h).f2(l) \tag{8}$$

The length function is a monotonically decreasing function with respect to the path length *l* between two concepts. This is captured by introducing a constant alpha.

$$\text{f2(l)} = e^{-\propto l} \tag{9}$$

Likewise, the depth function is monotonically increasing with respect to the depth *h* of concept in the hierarchy.

$$\text{f1(h)} = \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{10}$$

The similarity between two concepts is then calculated by:

$$S(w_1, w_2) = e^{-\propto l}.\frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{11}$$

Li et al. (2006) empirically discovered that for optimal performance in WordNet, alpha should be set to 0.2 and Beta set to 0.45. To aggregate the similarities, we used the formula in *equation 7* with a threshold of 0.25.

Vector Space based Similarity: We used the feature extraction module of the *scikit-learn* to extract the *TFIDF* weighted feature vectors for the two sentences and then calculated the cosine similarity between them:

$$\cos(A, B) = \frac{\sum_{t=1}^{n} TFIDF(w_{t,A})TFIDF(w_{t,B})}{\sqrt{\sum_{t=1}^{n} w_{t,A}^2 \sum_{t=1}^{n} w_{t,B}^2}} \tag{12}$$

Where *A* and *B* are the two texts being compared for similarity. The term frequency *TF* and inverse document frequency *IDF* are computed solely from the compared sentence pairs.

Compositional Distributional Approach: A limitation of Distributional Hypothesis (Harris, 1954; Firth, 1957) is that it captures the meaning of words in isolation. However, the true meaning of a sentence must take into account the interplay between the words it contains (Mitchell and Lapata, 2008; Mitchell and Lapata, 2009; Grefenstette et al., 2014). To account for this, we perform vector composition of the words in each sentence, using both additive and multiplicative composition (Mitchell and Lapata, 2010; Baroni, 2013). We obtained the vectors from the Word2Vec model described earlier. For the additive model, the vectors of all the words in a sentence are summed together to get a single vector for that sentence. Likewise, in the multiplicative model, vectors of all words in a sentence are multiplied component-wise to obtain a single vector for the sentence. To obtain similarity scores for each composition method, i.e., additive and multiplicative, we take the cosine of the vector space representation of the two sentences being compared.

3 System Description

For each sentence pair, our system generates 36 lexical, syntactic and semantic features. We experiment with using three distinct learning algorithms to map our feature representation onto an STS similarity score for the pair. The three learning algorithms correspond to the three runs we submitted to the shared task: ***Normas-RF1***, was trained with *Random Forest*

Dataset	Run1	Run2	Run3	Mean	Baseline-Median	Baseline-Best
Answer	.160	**.365**	.276	.267	.480	.692
Question	.467	.613	**.653**	.577	.571	.747
Postediting	.720*	**.802***	.797*	.773	.812	.866
Plagiarism	.621	**.746**	.724	.697	.789	.841
Headlines	.588	.688	**.722**	.666	.764	.827
Runs Mean	.508	.640	.630	.596		

Table 1: Summary of Pearson Correlation Evaluation Of The Submitted System

(Breiman, 2001; Liaw and Wiener, 2002); ***Normas-SV2***, uses *Support Vector Regression* (Chang and Lin, 2011; Basak et al., 2007); ***Normas-ECV3***, is based on *Elastic Net* (Zou and Hastie, 2005). As training data, we used ***2234*** sentence pairs from the *2012* training data. Parameter fine-tuning[7] for each of the algorithms was done using *OnWN* dataset of the STS 2013 evaluation data as the development test set. We used the scikit-learn[8] implementation of the three algorithms.

4 Evaluation and Discussion

We conducted two experiments, *Pre-Submission* and *Post-Submission* experiments. Table 1 summarizes the result of the Runs submitted across the datasets for the first experiment (*Pre-Submission*). The scores in **bold** shows the best scores per dataset. The scores with *asterisk (*)* appended shows the best scores under each Run. We used the preliminary result of the 2016 STS task released by the organizers as the baseline for evaluation. The preliminary result includes the median scores (*Baseline-Median*) and best scores (*Baseline-Best*) of the participating systems on each dataset. The *Baseline Best* is the score of the top performing system for each dataset of the Semeval 2016 task(Agirre et al., 2016).

It can be seen from *Table 1* that our best scores were from #Run2 and #Run3. The Random Forest algorithm performed poorly compared to the other two Runs on all of the datasets. All three Runs performed best on the Post-Editing dataset. This conforms to the pattern observed from the Baseline scores. Analysis of this dataset revealed that the sentences are longer and share more words.

Our worst performance across board is on the Answer-Answer dataset. Inspecting the data reveals that the sentences are both short and tend to share words that have little or no impact on their overall meaning. Our models may have been misguided by these spurious matching words.

Overall, our best systems, #Run2 and #Run3, have results very close to the *Baseline-Best* on three datasets and outclassed *Baseline-Median* on Question-question dataset. Our system performance may have been handicapped by the limited amount of data we used for training. Recall that we trained our system on only **2234** sentences of the 2012 training data. The tiny size of the dataset was necessitated by the complexity of computing some of the semantic features. Specifically, the WordNet similarity features used. Also, analysis of our training data shows that the sentences are of few words (short) and with high term overlap. Perhaps, our system could have performed better with more training data, especially if we had used a dataset with long sentences and also included more training data from previous STS evaluation tasks.

After the official evaluation, we reproduced our experiment using a larger training set. The new training data contains **9902** sentences drawn from the *2012-2014* evaluation datasets. Using more training data resulted in a notable improvement in performance. *Table 2* reports the results obtained by our (*post-submission*) systems. The *Percentage Gain* column shows the improvement in performance when the *Mean* result in the second experiment is compared to the one submitted initially (*pre-submission*).

The best improvement observed is from the *Answer-Answer* dataset with **38.0** % improvement

[7] For SVR, we used the LibSvm scikit implementation with RBF kernel. We set C=1.0, epsilon=0.2 and cache size=200. For Elastic Net we used alpha=0.5. Random Forest, we used 100 trees, max_depth=None and max_leaf_nodes=None. Grid search was used for hyperparameter optimization.

[8] http://scikit-learn.org/

Dataset	Run1	Run2	Run3	Mean	Percent Gain	Baseline-Median	Baseline-Best
Answer	.380	**.542**	.371	.431	38.0	.480	.692
Question	.698	.714	**.723**	.711	18.8	.571	.747
Postediting	.819*	**.852***	.809*	.826	6.80	.812	.866
Plagiarism	.797	**.821**	.784	.800	12.8	.789	.841
Headlines	**.805**	.789	.780	.791	15.8	.764	.827
Runs Mean	.699	.743	.693	.711	16.1	.683	.794

Table 2: Post-Submission Experiment With More Training Data (9902 sentence pairs)

across all Runs. The result from our top performing Runs across all datasets clearly surpass the *Baseline-Median* results. The top scores for each dataset were also very close to the *Baseline-Best* results. Our overall Mean of **0.711** surpasses the overall Mean of the *Baseline-Median (0.683)* and is far better than the 0.596 obtained by our submitted systems.

5 Conclusion

This paper has described our system submission to the SemEval 2016 STS shared task. We participated in the STS-Core (*Monolingual subtask*). We submitted three Runs using the same feature set but with models built by different machine learning algorithms. We obtained the best performance from the *Support Vector Regression* and *Elastic Net* based models and observed relatively poor performance from *Random Forests*.

The systems we submitted to the shared task obtained results that are very close to the best performing shared task system in 3 datasets and above the median mark on another dataset .

After the shared task, we reproduced our experiment on a bigger training dataset. We obtained a significant improvement (16.1%) when compared to the first experiment.

In future work, we plan to investigate the significance of all features so as to identify noisy or less important ones as well as to attempt to identify the optimal combination of features. Another improvement might be to build an *ensemble* combining individual systems built using variations in any one of the following: learning algorithm, feature set, or training data sampling (e.g., combining our best two systems).

References

Eugene Agichtein, Walt Askew, and Yandong Liu. 2008. Combining lexical, syntactic, and semantic evidence for textual entailment classification. *Proceedings of TAC*, 31.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In In* *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, USA, June. Association for Computational Linguistics.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 435–440. Association for Computational Linguistics.

Marco Baroni. 2013. Composition in distributional semantics. *Language and Linguistics Compass*, 7(10):511–522.

Regina Barzilay and Michael Elhadad. 1999. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121.

Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. 2007. Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):203–224.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Measuring semantic similarity between words using web search engines. *www*, 7:757–766.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Ido Dagan. 2000. Contextual word similarity. *Handbook of Natural Language Processing*, pages 459–475.

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

Tamer Elsayed, Jimmy Lin, and Douglas W Oard. 2008. Pairwise document similarity in large collections with mapreduce. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 265–268. Association for Computational Linguistics.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52. Citeseer.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

John R Firth. 1957. {A synopsis of linguistic theory, 1930-1955}.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25. ACM.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. 2014. Concrete sentence spaces for compositional distributional models of meaning. In *Computing Meaning*, pages 71–86. Springer.

Dan Gusfield. 1997. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Anna Huang. 2008. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56.

Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on*, 18(8):1138–1150.

Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomforest. *R news*, 2(3):18–22.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

Jeff Mitchell and Mirella Lapata. 2009. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 430–439. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 441–448. Association for Computational Linguistics.

Antonio Si, Hong Va Leong, and Rynson WH Lau. 1997. Check: a document plagiarism detection system. In *Proceedings of the 1997 ACM symposium on Applied computing*, pages 70–77. ACM.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.