UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems

Lushan Han, Abhay Kashyap, Tim Finin
Computer Science and
Electrical Engineering
University of Maryland, Baltimore County
Baltimore MD 21250
{lushan1,abhay1,finin}@umbc.edu

James Mayfield and Jonathan Weese
Human Language Technology
Center of Excellence
Johns Hopkins University
Baltimore MD 21211
mayfield@jhu.edu, jonny@cs.jhu.edu

Abstract

We describe three semantic text similarity systems developed for the *SEM 2013 STS shared task and the results of the corresponding three runs. All of them shared a word similarity feature that combined LSA word similarity and WordNet knowledge. The first, which achieved the best mean score of the 89 submitted runs, used a simple term alignment algorithm augmented with penalty terms. The other two runs, ranked second and fourth, used support vector regression models to combine larger sets of features.

1 Introduction

Measuring semantic text similarity has been a research subject in natural language processing, information retrieval and artificial intelligence for many years. Previous efforts have focused on comparing two long texts (e.g., for document classification) or a short text with a long text (e.g., Web search), but there are a growing number of tasks requiring computing the semantic similarity between two sentences or other short text sequences. They include paraphrase recognition (Dolan et al., 2004), Twitter tweets search (Sriram et al., 2010), image retrieval by captions (Coelho et al., 2004), query reformulation (Metzler et al., 2007), automatic machine translation evaluation (Kauchak and Barzilay, 2006) and schema matching (Han et al., 2012).

There are three predominant approaches to computing short text similarity. The first uses information retrieval's vector space model (Meadow, 1992) in which each text is modeled as a "bag of words" and represented using a vector. The similarity between two texts is then computed as the cosine similarity of the vectors. A variation on this approach leverages web search results (e.g., snippets) to provide context for the short texts and enrich their vectors using the words in the snippets (Sahami and Heilman, 2006). The second approach is based on the assumption that if two sentences or other short text sequences are semantically equivalent, we should be able to align their words or expressions. The alignment quality can serve as a similarity measure. This technique typically pairs words from the two texts by maximizing the summation of the word similarity of the resulting pairs (Mihalcea et al., 2006). The third approach combines different measures and features using machine learning models. Lexical, semantic and syntactic features are computed for the texts using a variety of resources and supplied to a classifier, which then assigns weights to the features by fitting the model to training data (Saric et al., 2012).

For evaluating different approaches, the 2013 Semantic Textual Similarity (STS) task asked automatic systems to compute sentence similarity according to a scale definition ranging from 0 to 5, with 0 meaning unrelated and 5 semantically equivalent (Agirre et al., 2012; Agirre et al., 2013). The example sentence pair "The woman is playing the violin" and "The young lady enjoys listening to the guitar" is scored as only *1* and the pair "The bird is bathing in the sink" and "Birdie is washing itself in the water basin" is given a score of *5*.

The vector-space approach tends to be too shallow for the task, since solving it well requires discriminating word-level semantic differences and goes be-

yond simply comparing sentence topics or contexts. Our first run uses an *align-and-penalize* algorithm, which extends the second approach by giving penalties to the words that are poorly aligned. Our other two runs use a support vector regression model to combine a large number of general and domain specific features. An important and fundamental feature used by all three runs is a powerful semantic word similarity model based on a combination of Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Landauer and Dumais, 1997) and knowledge from WordNet (Miller, 1995).

The remainder of the paper proceeds as follows. Section 2 presents the hybrid word similarity model. Section 3 describes the align-and-penalize approach used for the *PairingWords* run. In Section 4 we describe the SVM approach used for the *Galactus* and *Saiyan* runs. Section 5 discusses the results and is followed by a short conclusion.

2 Semantic Word Similarity Model

Our word similarity model was originally developed for the Graph of Relations project (UMBC, 2013a) which maps informal queries with English words and phrases for an RDF linked data collection into a SPARQL query. For this, we wanted a metric in which only the semantics of a word is considered and not its lexical category. For example, the verb "marry" should be semantically similar to the noun "wife". Another desiderata was that the metric should give highest scores and lowest scores in its range to similar and non-similar words, respectively. In this section, we describe how we constructed the model by combining LSA word similarity and WordNet knowledge.

2.1 LSA Word Similarity

LSA Word Similarity relies on the distributional hypothesis that words occurring in the same contexts tend to have similar meanings (Harris, 1968).

2.1.1 Corpus Selection and Processing

In order to produce a reliable word co-occurrence statistics, a very large and balanced text corpus is required. After experimenting with several corpus choices including Wikipedia, Project Gutenberg e-Books (Hart, 1997), ukWaC (Baroni et al., 2009), Reuters News stories (Rose et al., 2002) and LDC gigawords, we selected the Web corpus from the Stanford WebBase project (Stanford, 2001). We used the February 2007 crawl, which is one of the largest collections and contains 100 million web pages from more than 50,000 websites. The WebBase project did an excellent job in extracting textual content from HTML tags but still has abundant text duplications, truncated text, non-English text and strange characters. We processed the collection to remove undesired sections and produce high quality English paragraphs. We detected paragraphs using heuristic rules and only retrained those whose length was at least two hundred characters. We eliminated non-English text by checking the first twenty words of a paragraph to see if they were valid English words. We used the percentage of punctuation characters in a paragraph as a simple check for typical text. We removed duplicated paragraphs using a hash table. Finally, we obtained a three billion words corpus of good quality English, which is available at (Han and Finin, 2013).

2.1.2 Word Co-Occurrence Generation

We performed POS tagging and lemmatization on the WebBase corpus using the Stanford POS tagger (Toutanova et al., 2000). Word/term co-occurrences are counted in a moving window of a fixed size that scans the entire corpus[1]. We generated two co-occurrence models using window sizes ± 1 and ± 4 because we observed different natures of the models. ± 1 window produces a context similar to the dependency context used in (Lin, 1998a). It provides a more precise context but only works for comparing words within the same POS. In contrast, a context window of ± 4 words allows us to compute semantic similarity between words with different POS.

Our word co-occurrence models were based on a predefined vocabulary of more than 22,000 common English words and noun phrases. We also added to it more than 2,000 verb phrases extracted from WordNet. The final dimensions of our word co-occurrence matrices are 29,000 $\times$ 29,000 when words are POS tagged. Our vocabulary includes only open-class words (i.e. nouns, verbs, adjectives and adverbs). There are no proper nouns in the vocabulary with the only exception of country names.

[1] We used a stop-word list consisting of only the three articles "a", "an" and "the".

Word Pair	±4 model	±1 model
1. doctor_NN, physician_NN	0.775	0.726
2. car_NN, vehicle_NN	0.748	0.802
3. person_NN, car_NN	0.038	0.024
4. car_NN, country_NN	0.000	0.016
5. person_NN, country_NN	0.031	0.069
6. child_NN, marry_VB	0.098	0.000
7. wife_NN, marry_VB	0.548	0.274
8. author_NN, write_VB	0.364	0.128
9. doctor_NN, hospital_NN	0.473	0.347
10. car_NN, driver_NN	0.497	0.281

Table 1: Ten examples from the LSA similarity model

2.1.3 SVD Transformation

Singular Value Decomposition (SVD) has been found to be effective in improving word similarity measures (Landauer and Dumais, 1997). SVD is typically applied to a *word by document* matrix, yielding the familiar LSA technique. In our case we apply it to our *word by word* matrix. In literature, this variation of LSA is sometimes called HAL (Hyperspace Analog to Language) (Burgess et al., 1998).

Before performing SVD, we transform the raw word co-occurrence count f_{ij} to its log frequency $log(f_{ij} + 1)$. We select the 300 largest singular values and reduce the 29K word vectors to 300 dimensions. The LSA similarity between two words is defined as the cosine similarity of their corresponding word vectors after the SVD transformation.

2.1.4 LSA Similarity Examples

Ten examples obtained using LSA similarity are given in Table 1. Examples 1 to 6 illustrate that the metric has a good property of differentiating similar words from non-similar words. Examples 7 and 8 show that the ± 4 model can detect semantically similar words even with different POS while the ± 1 model yields much worse performance. Example 9 and 10 show that highly related but not substitutable words can also have a strong similarity but the ± 1 model has a better performance in discriminating them. We call the ± 1 model and the ± 4 model as *concept similarity* and *relation similarity* respectively since the ± 1 model has a good performance on nouns and the ± 4 model is good at computing similarity between relations regardless of POS of words, such as "marry to" and "is the wife of".

2.2 Combining with WordNet Knowledge

Statistical word similarity measures have limitations. Related words can have similarity scores as high as what similar words get, as illustrated by "doctor" and "hospital" in Table 1. Word similarity is typically low for synonyms having many word senses since information about different senses are mashed together (Han et al., 2013). By using WordNet, we can reduce the above issues.

2.2.1 Boosting LSA similarity using WordNet

We increase the similarity between two words if any of the following relations hold.

- They are in the same WordNet synset.
- One word is the direct hypernym of the other.
- One word is the two-link indirect hypernym of the other.
- One adjective has a direct *similar to* relation with the other.
- One adjective has a two-link indirect *similar to* relation with the other.
- One word is a derivationally related form of the other.
- One word is the head of the gloss of the other or its direct hypernym or one of its direct hyponyms.
- One word appears frequently in the glosses of the other and its direct hypernym and its direct hyponyms.

We use the algorithm described in (Collins, 1999) to find a word gloss header. We require a minimum LSA similarity of 0.1 between the two words to filter out noisy data when extracting WordNet relations.

We define a word's "significant senses" to deal with the problem of WordNet trivial senses. The word "year", for example, has a sense "a body of students who graduate together" which makes it a synonym of the word "class". This causes problems because "year" and "class" are not similar, in general. A sense is significant, if any of the following conditions are met: (i) it is the first sense; (ii) its WordNet frequency count is not less than five; or (iii) its word form appears first in its synset's word

form list and it has a WordNet sense number less than eight.

We assign path distance of zero to the category 1, path distance of one to the category 2, 4 and 6, and path distance of two to the other categories. The new similarity between word x and y by combining LSA similarity and WordNet relations is shown in the following equation

$$sim_{\oplus}(x, y) = sim_{LSA}(x, y) + 0.5e^{-\alpha D(x,y)} \quad (1)$$

where $D(x, y)$ is the minimal path distance between x and y. Using the $e^{-\alpha D(x,y)}$ to transform simple shortest path length has been demonstrated to be very effective according to (Li et al., 2003). The parameter α is set to be 0.25, following their experimental results. The ceiling of $sim_{\oplus}(x, y)$ remains 1.0 and we simply cut the excess.

2.2.2 Dealing with words of many senses

For a word w with many WordNet senses (currently ten or more), we use its synonyms with fewer senses (at most one third of that of w) as its substitutions in computing similarity with another word. Let S_x and S_y be the sets of all such substitutions of the words x and y respectively. The new similarity is obtained using Equation 2.

$$sim(x, y) = max(\max_{s_x \in S_x \cup \{x\}} sim_{\oplus}(s_x, y), \max_{s_y \in S_y \cup \{y\}} sim_{\oplus}(x, s_y)) \quad (2)$$

An online demonstration of a similar model developed for the GOR project is available (UMBC, 2013b), but it lacks some of this version's features.

3 Align-and-Penalize Approach

First we hypothesize that STS similarity between two sentences can be computed using

$$STS = T - P' - P'' \quad (3)$$

where T is the term alignments score, P' is the penalty for bad term alignments and P'' is the penalty for syntactic contradictions led by the alignments. However P'' had not been fully implemented and was not used in our STS submissions. We show it here just for completeness.

3.1 Aligning terms in two sentences

We start by applying the Stanford POS tagger to tag and lemmatize the input sentences. We use our predefined vocabulary, POS tagging data and simple regular expressions to recognize multi-word terms including noun and verb phrases, proper nouns, numbers and time. We ignore adverbs with frequency count larger than $500,000$ in our corpus and stop words with general meaning.

Equation 4 shows our aligning function g which finds the counterpart of term $t \in S$ in sentence S'.

$$g(t) = \underset{t' \in S'}{argmax}\ sim'(t, t') \quad (4)$$

$sim'(t, t')$ is a wrapper function over $sim(x, y)$ in Equation 2 that uses the *relation similarity* model. It compares numerical and time terms by their values. If they are equal, 1 is returned; otherwise 0. $sim'(t, t')$ provides limited comparison over pronouns. It returns 1 between subject pronouns *I*, *we*, *they*, *he*, *she* and their corresponding object pronouns. $sim'(t, t')$ also outputs 1 if one term is the acronym of the other term, or if one term is the head of the other term, or if two consecutive terms in a sentence match a single term in the other sentence (e.g. "long term" and "long-term"). $sim'(t, t')$ further adds support for matching words[2] not presented in our vocabulary using a simple string similarity algorithm. It computes character bigram sets for each of the two words without using padding characters. Dice coefficient is then applied to get the degree of overlap between the two sets. If it is larger than two thirds, $sim'(t, t')$ returns a score of 1; otherwise 0.

$g(t)$ is direction-dependent and does not achieve one-to-one mapping. This property is useful in measuring STS similarity because two sentences are often not exact paraphrases of one another. Moreover, it is often necessary to align multiple terms in one sentence to a single term in the other sentence, such as when dealing with repetitions and anaphora or, e.g., mapping "people writing books" to "writers".

Let S_1 and S_2 be the sets of terms in two input sentences. We define term alignments score T as the following equation shows.

$$\frac{\sum_{t \in S_1} sim'(t, g(t))}{2 \cdot |S_1|} + \frac{\sum_{t \in S_2} sim'(t, g(t))}{2 \cdot |S_2|} \quad (5)$$

[2]We use the regular expression "[A-Za-z][A-Za-z]*" to identify them.

3.2 Penalizing bad term alignments

We currently treat two kinds of alignments as "bad", as described in Equation 6. For the set B_i, we have an additional restriction that neither of the sentences has the form of a negation. In defining B_i, we used a collection of antonyms extracted from WordNet (Mohammad et al., 2008). Antonym pairs are a special case of disjoint sets. The terms "piano" and "violin" are also disjoint but they are not antonyms. In order to broaden the set B_i we will need to develop a model that can determine when two terms belong to disjoint sets.

$$A_i = \left\{\langle t, g(t)\rangle \,|t \in S_i \wedge sim'(t, g(t)) < 0.05\right\}$$
$$B_i = \{\langle t, g(t)\rangle \,|t \in S_i \wedge t \text{ is an antonym of } g(t)\}$$
$$i \in \{1, 2\} \quad (6)$$

We show how we compute P' in Equation 7.

$$P_i^A = \frac{\sum_{\langle t,g(t)\rangle \in A_i} (sim'(t, g(t)) + w_f(t) \cdot w_p(t))}{2 \cdot |S_i|}$$
$$P_i^B = \frac{\sum_{\langle t,g(t)\rangle \in B_i} (sim'(t, g(t)) + 0.5)}{2 \cdot |S_i|}$$
$$P' = P_1^A + P_1^B + P_2^A + P_2^B \quad (7)$$

The $w_f(t)$ and $w_p(t)$ terms are two weighting functions on the term t. $w_f(t)$ inversely weights the log frequency of term t and $w_p(t)$ weights t by its part of speech tag, assigning 1.0 to verbs, nouns, pronouns and numbers, and 0.5 to terms with other POS tags.

4 SVM approach

We used the scores from the align-and-penalize approach along with several other features to learn a support vector regression model. We started by applying the following preprocessing steps.

- The sentences were tokenized and POS-tagged using NLTK's (Bird, 2006) default Penn Treebank based tagger.
- Punctuation characters were removed from the tokens except for the decimal point in numbers.
- All numbers written as words were converted into numerals, e.g., "2.2 million" was replaced by "2200000" and "fifty six" by "56".
- All mentions of time were converted into military time, e.g., "5:40pm" was replaced by "1740" and "1h30am" by "0130".
- Abbreviations were expanded using a compiled list of commonly used abbreviations.
- About 80 stopwords were removed.

4.1 Ngram Matching

The sentence similarities are derived as a function of the similarity scores of their corresponding paired word ngrams. These features closely resemble the ones used in (Saric et al., 2012). For our system, we used unigrams, bigrams, trigrams and skip-bigrams, a special form of bigrams which allow for arbitrary distance between two tokens.

An ngram from the first sentence is exclusively paired with an ngram from the second which has the highest similarity score. Several similarity metrics are used to generate different features. For bigrams, trigrams and skip-bigrams, the similarity score for two ngrams is computed as the arithmetic mean of the similarity scores of the individual words they contain. For example, for the bigrams "he ate" and "she spoke", the similarity score is the average of the similarity scores between the words "he" and "she" and the words "ate" and "spoke".

The ngram overlap of two sentences is defined as "*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*" (Saric et al., 2012). Given sets S_1 and S_2 containing ngrams from sentences 1 and 2, and sets P_1 and P_2 containing their paired ngrams along with their similarity scores, the ngram overlap score for a given ngram type is computed using the following equation.

$$HM\left(\frac{\sum_{n \epsilon P_1} w(n).sim(n)}{\sum_{n \epsilon S_1} w(n)}, \frac{\sum_{n \epsilon P_2} w(n).sim(n)}{\sum_{n \epsilon S_2} w(n)}\right) \quad (8)$$

In this formula, HM is the harmonic mean, $w(n)$ is the weight assigned for the given ngram and $sim(n)$ is the similarity score of the paired word.

By default, all the ngrams are assigned a uniform weight of 1. But since different words carry different amount of information, e.g. "acclimatize" vs. "take", "cardiologist" vs. "person", we also use information content as weights. The information content of a word is as defined in (Saric et al., 2012).

$$ic(w) = ln\left(\frac{\sum_{w' \epsilon C} freq(w')}{freq(w)}\right) \quad (9)$$

Here C is the set of words in the corpus and $freq(w)$ is the frequency of a word in the corpus. The weight of an ngram is the sum of its constituent word weights. We use refined versions of Google ngram frequencies (Michel et al., 2011) from (Mem, 2008) and (Saric et al., 2012) to get the information content of the words. Words not in this list are assigned the average weight.

We used several word similarity metrics for ngram matching apart from the similarity metric described in section 2. Our baseline similarity metric was an exact string match which assigned a score of 1 if two tokens contained the same sequence of characters and 0 otherwise. We also used NLTK's library to compute WordNet based similarity measures such as Path Distance Similarity, Wu-Palmer Similarity (Wu and Palmer, 1994) and Lin Similarity (Lin, 1998b). For Lin Similarity, the Semcor corpus was used for the information content of words.

4.2 Contrast Scores

We computed contrast scores between two sentences using three different lists of antonym pairs (Mohammad et al., 2008). We used a large list containing 3.5 million antonym pairs, a list of about 22,000 antonym pairs from Wordnet and a list of 50,000 pairs of words with their degree of contrast. Contrast scores between two sentences were derived as a function of the number of antonym pairs between them similar to equation 8 but with negative values to indicate contrast scores.

4.3 Features

We constructed 52 features from different combinations of similarity metrics, their parameters, ngram types (unigram, bigram, trigram and skip-bigram) and ngram weights (equal weight vs. information content) for all sentence pairs in the training data.

- We used scores from the align-and-penalize approach directly as a feature.
- Using exact string match over different ngram types and ngram weights, we extracted eight features ($4 * 4$). We also developed four additional features ($2 * 2$) by includin stopwords in bigrams and trigrams, motivated by the nature of MSRvid dataset.
- We used the LSA boosted similarity metric in three modes: concept similarity, relation similarity and mixed mode, which used the concept model for nouns and relation model for verbs, adverbs and adjectives. A total of 24 features were extracted ($4 * 2 * 3$).
- For Wordnet-based similarity measures, we used uniform weights for Path and Wu-Palmer similarity and used the information content of words (derived from the Semcor corpus) for Lin similarity. Skip bigrams were ignored and a total of nine features were produced ($3 * 3$).
- Contrast scores used three different lists of antonym pairs. A total of six features were extracted using different weight values ($3 * 2$).

4.4 Support Vector Regression

The features described in 4.3 were used in different combinations to train several support vector regression (SVR) models. We used LIBSVM (Chang and Lin, 2011) to learn the SVR models and ran a grid search provided by (Saric et al., 2012) to find the optimal values for the parameters C, g and p. These models were then used to predict the scores for the test sets.

The *Galactus* system was trained on all of STS 2012 data and used the full set of 52 features. The FnWN dataset was handled slightly differently from the others. We observed that terms like "frame" and "entity" were used frequently in the five sample sentence pairs and treated them as stopwords. To accommodate the vast difference in sentence lengths, equation 8 was modified to compute the arithmetic mean instead of the harmonic mean.

The *Saiyan* system employed data-specific training and features. The training sets were subsets of the supplied STS 2012 dataset. More specifically, the model for headlines was trained on 3000 sentence pairs from MSRvid and MSRpar, SMT used 1500 sentence pairs from SMT europarl and SMT news, while OnWN used only the 750 OnWN sentence pairs from last year. The FnWN scores were directly used from the Align-and-Penalize approach. None of the models for *Saiyan* used contrast features and the model for SMT also ignored similarity scores from exact string match metric.

5 Results and discussion

Table 2 presents the official results of our three runs in the 2013 STS task. Each entry gives a run's Pearson correlation on a dataset as well as the rank of the run among all 89 runs submitted by the 35 teams. The last row shows the mean of the correlations and the overall ranks of our three runs.

We tested performance of the align-and-penalize approach on all of the 2012 STS datasets. It obtained correlation values of 0.819 on MSRvid, 0.669 on MSRpar, 0.553 on SMTeuroparl, 0.567 on SMTnews and 0.722 on OnWN for the test datasets, and correlation values of 0.814 on MSRvid, 0.707 on MSRpar and 0.646 on SMTeuroparl for the training datasets. The performance of the approach without using the antonym penalty is also tested, producing correlation scores of 0.795 on MSRvid, 0.667 on MSRpar, 0.554 on SMTeuroparl, 0.566 on SMTnew and 0.727 on OnWN, for the test datasets, and 0.794 on MSRvid, 0.707 on MSRpar and 0.651 on SMTeuroparl for the training datasets. The average of the correlation scores on all eight datasets with and without the antonym penalty is 0.6871 and 0.6826, respectively. Since the approach's performance was only slightly improved when the antonym penalty was used, we decided to not include this penalty in our *PairingWords* run in the hope that its simplicity would make it more robust.

During development, our SVM approach achieved correlations of 0.875 for MSRvid, 0.699 for MSRpar, 0.559 for SMTeuroparl, 0.625 for SMTnews and 0.729 for OnWN on the 2012 STS test data. Models were trained on their respective training sets while SMTnews used SMTeuroparl and OnWN used all the training sets. We experimented with different features and training data to study their influence on the performance of the models. We found that the unigram overlap feature, based on boosted LSA similarity and weighted by information content, could independently achieve very high correlations. Including more features improved the accuracy slightly and in some cases added noise. The difficulty in selecting data specific features and training for novel datasets is indicated by *Saiyan's* contrasting performance on headlines and OnWN datasets. The model used for Headlines was trained on data from seemingly different domains (MSRvid, MSRpar) while OnWN was trained only on OnWN from STS 2012. When the model for headlines dataset was used to predict the scores for OnWN, the correlation jumped from 0.55 to 0.71 indicating that the earlier model suffered from overfitting.

Dataset	Pairing	Galactus	Saiyan
Headlines (750 pairs)	0.7642 (3)	0.7428 (7)	**0.7838 (1)**
OnWN (561 pairs)	0.7529 (5)	0.7053 (12)	0.5593 (36)
FNWN (189 pairs)	**0.5818 (1)**	0.5444 (3)	0.5815 (2)
SMT (750 pairs)	0.3804 (8)	0.3705 (11)	0.3563 (16)
Weighted mean	**0.6181 (1)**	0.5927 (2)	0.5683 (4)

Table 2: Performance of our three systems on the four test sets.

Overfitting is not evident in the performance of *PairingWords* and *Galactus*, which have more consistent performance over all datasets. The relatively simple *PairingWords* system has two advantages: it is faster, since the current *Galactus* requires computing a large number of features; and its performance is more predictable, since training is not needed thus eliminating noise induced from diverse training sets.

6 Conclusion

We described three semantic text similarity systems developed for the *SEM 2013 STS shared task and the results of the corresponding three runs we submitted. All of the systems used a lexical similarity feature that combined POS tagging, LSA word similarity and WordNet knowledge.

The first run, which achieved the best mean score out of all 89 submissions, used a simple term alignment algorithm augmented with two penalty metrics. The other two runs, ranked second and fourth out of all submissions, used support vector regression models based on a set of more than 50 additional features. The runs differed in their feature sets, training data and procedures, and parameter settings.

Acknowledgments

This research was supported by AFOSR award FA9550-08-1-0265 and a gift from Microsoft.

References

[Agirre et al.2012] 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*, pages 385–393. Association for Computational Linguistics.

[Agirre et al.2013] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In **SEM 2013: The Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics.

[Baroni et al.2009] M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

[Bird2006] Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL '06, pages 69–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Burgess et al.1998] C. Burgess, K. Livesay, and K. Lund. 1998. Explorations in context space: Words, sentences, discourse. *Discourse Processes*, 25:211–257.

[Chang and Lin2011] 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.

[Coelho et al.2004] T.A.S. Coelho, Pável Pereira Calado, Lamarque Vieira Souza, Berthier Ribeiro-Neto, and Richard Muntz. 2004. Image retrieval using multiple evidence ranking. *IEEE Trans. on Knowl. and Data Eng.*, 16(4):408–417.

[Collins1999] Michael John Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

[Deerwester et al.1990] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

[Dolan et al.2004] Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04. Association for Computational Linguistics.

[Ganitkevitch et al.2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *HLT-NAACL 2013*.

[Han and Finin2013] Lushan Han and Tim Finin. 2013. UMBC webbase corpus. http://ebiq.org/r/351.

[Han et al.2012] Lushan Han, Tim Finin, and Anupam Joshi. 2012. Schema-free structured querying of dbpedia data. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2090–2093. ACM.

[Han et al.2013] Lushan Han, Tim Finin, Paul McNamee, Anupam Joshi, and Yelena Yesha. 2013. Improving Word Similarity by Augmenting PMI with Estimates of Word Polysemy. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1307–1322.

[Harris1968] Zellig Harris. 1968. *Mathematical Structures of Language*. Wiley, New York, USA.

[Hart1997] M. Hart. 1997. Project gutenberg electronic books. http://www.gutenberg.org/wiki/Main_Page.

[Kauchak and Barzilay2006] David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *HLT-NAACL '06*, pages 455–462.

[Landauer and Dumais1997] T. Landauer and S. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. In *Psychological Review, 104*, pages 211–240.

[Li et al.2003] Y. Li, Z.A. Bandar, and D. McLean. 2003. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882.

[Lin1998a] Dekang Lin. 1998a. Automatic retrieval and clustering of similar words. In *Proc. 17th Int. Conf. on Computational Linguistics*, pages 768–774, Montreal, CN.

[Lin1998b] Dekang Lin. 1998b. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Meadow1992] Charles T. Meadow. 1992. *Text Information Retrieval Systems*. Academic Press, Inc.

[Mem2008] 2008. Google word frequency counts. http://bit.ly/10JdTRz.

[Metzler et al.2007] Donald Metzler, Susan Dumais, and Christopher Meek. 2007. Similarity measures for short segments of text. In *Proceedings of the 29th European conference on IR research*, pages 16–27. Springer-Verlag.

[Michel et al.2011] Jean-Baptiste Michel, Yuan K. Shen, Aviva P. Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker,

Martin A. Nowak, and Erez L. Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182, January 14.

[Mihalcea et al.2006] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence*, pages 775–780. AAAI Press.

[Miller1995] G.A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):41.

[Mohammad et al.2008] Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word-pair antonymy. In *Proc. Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-2008)*, October.

[Rose et al.2002] Tony Rose, Mark Stevenson, and Miles Whitehead. 2002. The reuters corpus volume 1 - from yesterdays news to tomorrows language resources. In *In Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 29–31.

[Sahami and Heilman2006] Mehran Sahami and Timothy D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 377–386. ACM.

[Saric et al.2012] Frane Saric, Goran Glavas, Mladen Karan, Jan Snajder, and Bojana Dalbelo Basic. 2012. Takelab: systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 441–448. Association for Computational Linguistics.

[Sriram et al.2010] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 2010. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM.

[Stanford2001] Stanford. 2001. Stanford WebBase project. http://bit.ly/WebBase.

[Toutanova et al.2000] Kristina Toutanova, Dan Klein, Christopher Manning, William Morgan, Anna Rafferty, and Michel Galley. 2000. Stanford log-linear part-of-speech tagger. http://nlp.stanford.edu/software/tagger.shtml.

[UMBC2013a] UMBC. 2013a. Graph of relations project. http://ebiq.org/j/95.

[UMBC2013b] UMBC. 2013b. Semantic similarity demonstration. http://swoogle.umbc.edu/SimService/.

[Wu and Palmer1994] Z. Wu and M. Palmer. 1994. Verb semantic and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-1994)*, pages 133–138, Las Cruces (Mexico).