DLS@CU-CORE: A Simple Machine Learning Model of Semantic Textual Similarity

Md. Arafat Sultan, Steven Bethard, Tamara Sumner
Institute of Cognitive Science and Department of Computer Science
University of Colorado, Boulder, CO 80309
{arafat.sultan, steven.bethard, sumner}@colorado.edu

Abstract

We present a system submitted in the Semantic Textual Similarity (STS) task at the Second Joint Conference on Lexical and Computational Semantics (*SEM 2013). Given two short text fragments, the goal of the system is to determine their semantic similarity. Our system makes use of three different measures of text similarity: word *n*-gram overlap, character *n*-gram overlap and semantic overlap. Using these measures as features, it trains a support vector regression model on SemEval STS 2012 data. This model is then applied on the STS 2013 data to compute textual similarities. Two different selections of training data result in very different performance levels: while a correlation of 0.4135 with gold standards was observed in the official evaluation (ranked 63rd among all systems) for one selection, the other resulted in a correlation of 0.5352 (that would rank 21st).

1 Introduction

Automatically identifying the semantic similarity between two short text fragments (e.g. sentences) is an important research problem having many important applications in natural language processing, information retrieval, and digital education. Examples include automatic text summarization, question answering, essay grading, among others.

However, despite having important applications, semantic similarity identification at the level of short text fragments is a relatively recent area of investigation. The problem was formally brought to attention and the first solutions were proposed in 2006 with the works reported in (Mihalcea et al., 2006) and (Li et al., 2006). Work prior to these focused primarily on large documents (or individual words) (Mihalcea et al., 2006). But the sentence-level granularity of the problem is characterized by factors like high specificity and low topicality of the expressed information, and potentially small lexical overlap even between very similar texts, asking for an approach different from those that were designed for larger texts.

Since its inception, the problem has seen a large number of solutions in a relatively small amount of time. The central idea behind most solutions is the identification and alignment of semantically similar or related words across the two sentences, and the aggregation of these similarities to generate an overall similarity score (Mihalcea et al., 2006; Islam and Inkpen, 2008; Šarić et al., 2012).

The Semantic Textual Similarity task (STS) organized as part of the Semantic Evaluation Exercises (see (Agirre et al., 2012) for a description of STS 2012) provides a common platform for evaluation of such systems via comparison with human-annotated similarity scores over a large dataset.

In this paper, we present a system which was submitted in STS 2013. Our system is based on very simple measures of lexical and character-level overlap, semantic overlap between the two sentences based on word relatedness measures, and surface features like the sentences' lengths. These measures are used as features for a support vector regression model that we train with annotated data from SemEval STS 2012. Finally, the trained model is applied on the STS 2013 test pairs.

Our approach is inspired by the success of similar systems in STS 2012: systems that combine multiple measures of similarity using a machine learning model to generate an overall score (Bär et al., 2012; Šarić et al., 2012). We wanted to investigate how a minimal system of this kind, making use of very few external resources, performs on a large dataset. Our experiments reveal that the performance of such a system depends highly on the training data. While training on one dataset yielded a best

*Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task*, pages 176–180, Atlanta, Georgia, June 13-14, 2013.

correlation (among our three runs, described later in this document) of only 0.4135 with the gold scores, training on another dataset showed a considerably higher correlation of 0.5352.

2 Computation of Text Similarity: System Overview

In this section, we present a high-level description of our system. More details on extraction of some of the measures of similarity are provided in Section 3.

Given two input sentences S_1 and S_2, our algorithm can be described as follows:

1. Compute semantic overlap (8 features):
   a. Lemmatize S_1 and S_2 using a memory-based lemmatizer[1] and remove all stop words.
   b. Compute the degree to which the concepts in S_1 are covered by semantically similar concepts in S_2 and vice versa (see Section 3 for details). The result of this step is two different 'degree of containment' values (S_1 in S_2 and vice versa).
   c. Compute the minimum, maximum, arithmetic mean and harmonic mean of the two values to use as features in the machine learning model.
   d. Repeat steps 1a through 1c for a weighted version of semantic overlap where each word in the first sentence is assigned a weight which is proportional to its specificity in a selected corpus (see Section 3).
2. Compute word n-gram overlap (16 features):
   a. Extract n-grams (for $n = 1, 2, 3, 4$) of all words in S_1 and S_2 for four different setups characterized by the four different value combinations of the two following variables: lemmatization (*on* and *off*), stopWordsRemoved (*on* and *off*).
   b. Compute the four measures (min, max, arithmetic and harmonic mean) for each value of *n*.
3. Compute character n-gram overlap (16 features):
   a. Repeat all steps in 2 above for character n-grams ($n = 2, 3, 4, 5$).
4. Compute sentence length features (2 features):
   a. Compute the lengths of S_1 and S_2; and the minimum and maximum of the two values.
   b. Include the ratio of the maximum to the minimum and the difference between the maximum and minimum in the feature set.
5. Train a support vector regression model on the features extracted in steps 1 through 4 above using data from SemEval 2012 STS (see Section 4 for specifics on the dataset). We used the LibSVM implementation of SVR in WEKA.
6. Apply the model on STS 2013 test data.

3 Semantic Overlap Measures

In this section, we describe the computation of the two sets of semantic overlap measures mentioned in step 1 of the algorithm in Section 2.

We compute semantic overlap between two sentences by first computing the semantic relatedness among their constituent words. Automatically computing the semantic relatedness between words is a well-studied problem and many solutions to the problem have been proposed. We compute word relatedness in two forms: semantic relatedness and string similarity. For semantic relatedness, we utilize two web services. The first one concerns a resource named ConceptNet (Liu and Singh, 2004), which holds a large amount of common sense knowledge concerning relationships between real-world entities. It provides a web service[2] that generates word relatedness scores based on these relationships. We will use the term $CNrel(w_1, w_2)$ to denote the relatedness of the two words w_1 and w_2 as generated by ConceptNet.

We also used the web service[3] provided by another resource named Wikipedia Miner (Milne and Witten, 2013). While ConceptNet successfully captures common sense knowledge about words and concepts, Wikipedia Miner specializes in identifying relationships between scientific concepts powered by Wikipedia's vast repository of scientific information (for example, *Einstein* and relativity). We will use the term $WMrel(w_1, w_2)$ to denote the relatedness of the two words w_1 and w_2 as generated by Wikipedia Miner. Using two systems enabled us

[1] http://www.clips.ua.ac.be/pages/MBSP#lemmatizer

[2] http://conceptnet5.media.mit.edu/data/5.1/assoc/c/en/cat? filter=/c/en/dog&limit=1

[3] http://wikipedia-miner.cms.waikato.ac.nz/services/compare? term1=cat&term2=dog

to increase the coverage of our word similarity computation algorithm.

Each of these web services return a score in the range [0, 1] where 0 represents no relatedness and 1 represents complete similarity. A manual inspection of both services indicates that in almost all cases where the services' word similarity scores deviate from what would be the human-perceived similarity, they generate lower scores (i.e. lower than the human-perceived score). This is why we take the maximum of the two services' similarity scores for any given word pair as their semantic relatedness:

$$semRel(w_1, w_2) = \max\{CNrel(w_1, w_2), WMrel(w_1, w_2)\}$$

We also compute the string similarity between the two words by taking a weighted combination of the normalized lengths of their longest common substring, subsequence and prefix (normalization is done for each of the three by dividing its length with the length of the smaller word). We will refer to the string similarity between words w_1 and w_2 as $stringSim(w_1, w_2)$. This idea is taken from (Islam and Inkpen, 2008); the rationale is to be able to find the similarity between (1) words that have the same lemma but the lemmatizer failed to lemmatize at least one of the two surface forms successfully, and (2) words at least one of which has been misspelled. We take the maximum of the string similarity and the semantic relatedness between two words as the final measure of their similarity:

$$sim(w_1, w_2) = \max\{semRel(w_1, w_2), stringSim(w_1, w_2)\}$$

At the sentence level, our first set of semantic overlap measures (step 1b) is an unweighted measure that treats all content words equally. More specifically, after the preprocessing in step 1a of the algorithm, we compute the degree of semantic coverage of concepts expressed by individual content words in S_1 by S_2 using the following equation:

$$cov_{uw}(S_1, S_2) = \frac{\sum_{s \in S_1} \left[\max_{t \in S_2}\{sim(s,t)\}\right]}{|S_1|}$$

where $sim(s,t)$ is the similarity between the two lemmas s and t.

We also compute a weighted version of semantic coverage (step 1d in the algorithm) by incorporating the specificity of each word (measured by its *information content*) as shown in the equation below:

$$cov_w(S_1, S_2) = \frac{\sum_{s \in S_1} \left[\max_{t \in S_2}\{ic(s).sim(s,t)\}\right]}{|S_1|}$$

where $ic(w)$ stands for the information content of the word w. Less common words (across a selected corpus) have high information content:

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

where *C* is the set of all words in the chosen corpus and *f(w)* is the frequency of the word *w* in the corpus. We have used the Google Unigram Corpus[4] to assign the required frequencies to these words.

4 Evaluation

The STS 2013 test data consists of four datasets: two datasets consisting of gloss pairs (*OnWN*: 561 pairs and *FNWN*: 189 pairs), a dataset of machine translation evaluation pairs (*SMT*: 750 pairs) and a dataset consisting of news headlines (*headlines*: 750 pairs). For each dataset, the output of a system is evaluated via comparison with human-annotated similarity scores and measured using the Pearson Correlation Coefficient. Then a weighted sum of the correlations for all datasets are taken to be the final score, where each dataset's weight is the proportion of sentence pairs in that dataset.

We computed the similarity scores using three different feature sets (for our three runs) for the support vector regression model:

1. All features mentioned in Section 2. This set of features were used in our run 1.
2. All features except word *n*-gram overlap (experiments on STS 2012 test data revealed that using word *n*-grams actually lowers the performance of our model, hence this decision). These are the features that were used in our run 2.
3. Only character *n*-gram and length features (just to test the performance of the model without

[4] http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

any semantic features). Our run 3 was based on these features.

We trained the support vector regression model on two different training datasets, both drawn from STS 2012 data:

1. In the first setup, we chose the training datasets from STS 2012 that we considered the most similar to the test dataset. The only exception was the *FNWN* dataset, for which we selected the all the datasets from 2012 because no single dataset from STS 2012 seemed to have similarity with this dataset. For the *OnWN* test dataset, we selected the *OnWN* dataset from STS 2012. For both *headlines* and *SMT*, we selected *SMT-news* and *SMTeuroparl* from STS 2012. The rationale behind this selection was to train the machine learning model on a distribution similar to the test data.
2. In the second setup, we aggregated all datasets (train and test) from STS 2012 and used this combined dataset to train the three models that were later applied on each STS 2013 test data. Here the rationale is to train on as much data as possible.

Table 1 shows the results for the first setup. This is the performance of the set of scores which we actually submitted in STS 2013. The first four columns show the correlations of our system with the gold standard for all runs. The rightmost column shows the overall weighted correlations. As we can see, run 1 with all the features demonstrated the best performance among the three runs. There was a considerable drop in performance in run 3 which did not utilize any semantic similarity measure.

Table 1. Results for manually selected training data

Run	headlines	OnWN	FNWN	SMT	Total
1	.4921	.3769	.4647	.3492	.4135
2	.4669	.4165	.3859	.3411	.4056
3	.3867	.2386	.3726	.3337	.3309

As evident from the table, evaluation results did not indicate a particularly promising system. Our best system ranked 63rd among the 90 systems evaluated in STS 2013. We further investigated to find out the reason: is the set of our features insufficient to capture text semantic similarity, or were the training data inappropriate for their corresponding test data? This is why we experimented with the second setup discussed above. Following are the results:

Table 2. Results for combined training data

Run	headlines	OnWN	FNWN	SMT	Total
1	.6854	.5981	.4647	.3518	.5339
2	.7141	.5953	.3859	.349	.5352
3	.6998	.4826	.3726	.3365	.4971

As we can see in Table 2, the correlations for all feature sets improved by more than 10% for each run. In this case, the best system with correlation 0.5352 would rank 21st among all systems in STS 2013. These results indicate that the primary reason behind the system's previous bad performance (Table 1) was the selection of an inappropriate dataset. Although it was not clear in the beginning which of the two options would be the better, this second experiment reveals that selecting the largest possible dataset to train is the better choice for this dataset.

5 Conclusions

In this paper, we have shown how simple measures of text similarity using minimal external resources can be used in a machine learning setup to compute semantic similarity between short text fragments. One important finding is that more training data, even when drawn from annotations on different sources of text and thus potentially having different feature value distributions, improve the accuracy of the model in the task. Possible future expansion includes use of more robust concept alignment strategies using semantic role labeling, inclusion of structural similarities of the sentences (e.g. word order, syntax) in the feature set, incorporating word sense disambiguation and more robust strategies of concept weighting into the process, among others.

References

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 First Joint Conference on Lexical and Computational Semantics*. ACL, Stroudsburg, PA, USA, 385-393.

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*. ACL, Stroudsburg, PA, USA, 435-440.

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data* 2, 2, Article 10 (July 2008), 25 pages.

Yuhua Li, David Mclean, Zuhair A. Bandar, James D. O'Shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, vol.18, no.8, 1138-1150.

Hugo Liu and Push Singh. 2004. ConceptNet — a practical commonsense reasoning tool-kit. *BT Technology Journal* 22, 4 (October 2004), 211-226.

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 - Volume 1* (AAAI'06), Anthony Cohn (Ed.), Vol. 1. AAAI Press 775-780.

David Milne and Ian H. Witten. 2013. An open-source toolkit for mining Wikipedia. *Artif. Intell.* 194 (January 2013), 222-239.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić.Šarić. 2012. TakeLab: systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*. ACL, Stroudsburg, PA, USA, 441-448.