Semantic textual similarity

Kristijan Bišćanić, Karlo Dumbović, Nino Jagar

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{kristijan.biscanic,karlo.dumbovic,nino.jagar}@fer.hr

Abstract

This paper focuses on building a system that can assess the semantic similarity between two texts. Each text consists of only one sentence, so more sophisticated approach should be used than just simple word overlap. The sentences are first being preprocessed and then specific features are extracted. The features are being used to train a support vector regression model. The model outputs similarity score which is then compared with human similarity judgements. The performance of the model is being evaluated using Pearson and Spearman correlation coefficients.

1. Introduction

Semantic similarity is a metric defined over a set of documents or terms, where the idea of distance between them is based on likeness of their meaning or semantic content as opposed to similarity which can be estimated regarding their syntactical representation (Wikipedia, 2015).

We have implemented a system that can measure the semantic similarity between two sentences whose context is previously unknown and of no matter to us. The idea of building such system comes from the necessity of automated similarity estimating in a number of study fields such as biomedical informatics, geoinformatics, linguistics and natural language processing (NLP).

The main task of the system is to compute certain features which act as a representation of sentence similarity. Each sentence is first preprocessed and then we compute various features from them. Exact features used are: ngram overlap, WordNet-augmented overlap, weighted word overlap, vector-space similarity, 2 types of normalized differences, shallow NERC and numbers overlap. Calculation of the features will be briefly described in forthcoming sections. These features are used as an input to a support vector regression (SVR) model. The model is being hyper-optimized by grid search over different values of hyper-parameters: regularization cost parameter C, RBF precision γ and penalty threshold ϵ. On its output, the model gives us the similarity judgement based on the features extracted. The similarity is scored as a real number from 0 to 5, where 0 represents no similarity, while 5 represents maximum similarity. Model estimations are compared with human judgements, and the accuracy of the model is measured using Pearson and Spearman correlation coefficients.

2. Overview of the field

Semantic textual similarity is an interesting field with an increasing interest in recent years. Various systems and algorithms are submitted in recent years for the Semeval/*SEM tasks organized in 2012, 2013 and 2014, with more than 60 participating teams (SemEval, 2015). We designed our system for the SemEval-2012 Task 6 (Agirre et al., 2012).

One implementation of similar system is discussed at length in (Šarić et al., 2012). The system described there directly inspired the implementation of our system.

3. Description of the System

We used a Support Vector Regression model (SVR) as our learning model. Our system first does a preprocessing step, and then we compute various features from preprocessed sentences.

3.1. Preprocessing

To make our system more robust to small differences in inputs, we use the following preprocessing steps on each sentence:

1. All dashes, brackets, slashes and hyphens are stripped;
2. Various quotes are replaced with regular *'* and *"*;
3. Words are lowercased for calculation of most features;
4. Names of currencies are stripped from the values, e.g., *€EUR100* becomes *€100*;
5. Words are tokenized using the pre-trained NLTK Punkt tokenizer;
6. Tokens *'m*, *n't* and *'re* are changed to *am*, *not* and *are*, respectively;
7. If a compound appears in one sentence, and it also appears in the other sentence but as two consecutive words, then they are replaced by that compound. E.g., *foot ball* from first sentence will be replaced with word *football* if it appears in the other sentence;
8. Words are POS-tagged using the Maxent Treebank POS-tagger from NLTK;
9. For calculation of some features, we removed stopwords using the NLTK stopwords corpus;
10. We performed lemmatization for calculation of some features using the WordNet corpus from NLTK.

3.2. Ngram Overlap

The ngram overlap between two sentences is the harmonic mean of the degree to which the first sentence overlaps with the second and the degree to which the second sentence overlaps with the first.

First we compute S_1 and S_2, sets of consecutive ngrams from first and second sentence, respectively. The ngram overlap is then calculated using the following equation:

$$overlap(S_1, S_2) = 2 \cdot \frac{|S_1 \cap S_2|}{|S_1| + |S_2|} \quad (1)$$

Our system uses overlap scores based on unigrams, bigrams and trigrams. We calculated the overlap on both regular words and lemmas.

3.3. Weighted word overlap

First, for every word w in corpus C we have to compute its information content ic using the following equation:

$$ic(w) = \ln \frac{\sum_{w' \in C} freq(w')}{freq(w)} \quad (2)$$

where $freq(w)$ is the word frequency that we obtain indirectly from Google Books Ngrams, whose coverage for English is extremely good. (Michel et al., 2011)

Then, we introduce the weighted word coverage of the second sentence by the first one, which is calculated using the following equation:

$$wwc(S_1, S_2) = \frac{\sum_{w \in S_1 \cap S_2} ic(w)}{\sum_{w \in S_2} ic(w)} \quad (3)$$

where S_1 and S_2 are sets of words appearing in the sentences.

Finally, the weighted word overlap is calculated as a harmonic mean of the $wwc(S_1, S_2)$ and $wwc(S_2, S_1)$.

3.4. Normalized differences

Normalized difference for two integers a and b is calculated as

$$ND(a, b) = \frac{|a - b|}{\max\{a, b\}}. \quad (4)$$

In our system, we use two types of normalized differences. The first one takes as input sentence lengths, while the second one uses aggregate information content of the words in sentences.

3.5. Numbers overlap

Inspired by , we decided to compute few features which will represent numbers overlap. To begin, we need to extract the numbers from each sentence in separate sets N_1 and N_2. The features we observe are computed as follows:

$$NO^{(1)} = \ln(1 + |N_1| + |N_2|) \quad (5)$$

$$NO^{(2)} = \frac{2|N_1 \cap N_2|}{|N_1| + |N_2|} \quad (6)$$

$$NO^{(3)} = (N_1 \subset N_2) \vee (N_2 \subset N_1) \quad (7)$$

As proposed in (Šarić et al., 2012), numbers that differ only in the number of decimal places are treated as equal.

3.6. WordNet-Augmented Word Overlap

The *WordNet-augmented word overlap* feature is defined as a harmonic mean of *WordNet-augmented coverages* $P_{WN}(S_1, S_2)$ and $P_{WN}(S_2, S_1)$. $P_{WN}(S_1, S_2)$ is computed as:

$$P_{WN}(S_1, S_2) = \frac{1}{|S_2|} \sum_{w \in S_1} score(w, S_2) \quad (8)$$

$$score(w, S) = \begin{cases} 1 & \text{if } w \in S \\ \max\limits_{w' \in S} sim(w, w') & \text{otherwise} \end{cases} \quad (9)$$

$sim(w, w')$ represents the WordNet path length similarity of words w and w'. We use this feature to find unigram overlap between sentences that are semantically similar but don't use exactly the same words. We are using WordNet to assign partial scores these words and thus allowing for some lexical variation.

3.7. Vector Space Sentence Similarity

We represent each sentence in the vector space as a single vector u. u is calculated by summing the LSA vector of each word in the sentence:

$$u(S) = \sum_{w \in S} \mathbf{x}_w \quad (10)$$

$\mathbf{x}_w$ is a word vector taken from pre-trained vectors file published by Google trained on part of Google News dataset. After calculating the representations of both sentences we define our feature as:

$$vsss(S_1, S_2) = |\cos(u(S_1), u(S_2))| \quad (11)$$

Another similar vector space representation u_W uses the information content $ic(w)$ of LSA vector in the following way:

$$u_W(S) = \sum_{w \in S} ic(w)\mathbf{x}_w \quad (12)$$

We define the second feature as:

$$vsss'(S_1, S_2) = |\cos(u_W(S_1), u_W(S_2))| \quad (13)$$

3.8. Shallow Named Entity Recognizer and Classifier

Shallow named entity recognizer and classifier treats every capitalized word as a named entity if it is longer than one character and classifies every word written in all caps and begging with a period as a stock index symbol. Our system uses four features associated with named entities: the overlap of named entities, the feature indicating if named entities were found in either of the sentences, the overlap of stock index symbols and the feature indicating if stock index symbols were found in either of the sentences.

4. Results

4.1. Model Training

We used LIBSVM (Chang and Lin, 2011) to train a separate SVR model for each training set. The model was hyper-optimized (in terms of Pearson correlation) by grid search with nested cross-validation ($k = 10$) to find the optimal parameters C, γ and ϵ. Final prediction results are then trimmed to a $[0, 5]$ interval. For the surprise test set *SMT-news* we trained our system on *SMTeuroparl* train set, and for the *OnWN* test set we trained the system on the union of all provided train sets. The performance on train sets is shown in Table 1.

TODO - koji su dobri/losi featurei?

Table 1: Cross-validated results on train sets

Set	Pearson	Spearman	C	γ	ϵ
MSRvid	1.0000	1.0000	1	1	1
MSRpar	1.0000	1.0000	1	1	1
SMTeuroparl	1.0000	1.0000	1	1	1

4.2. Test Set Results

We evaluated our model using Pearson and Spearman correlation coefficients. The performance on test sets is shown in Table 2. Aggregate performance according to three aggregate evaluation measures proposed in (Agirre et al., 2012) are shown in Table 3.

Table 2: Performance on test sets

Set	Pearson	Spearman
MSRvid	1.0000	1.0000
MSRpar	1.0000	1.0000
SMTeuroparl	1.0000	1.0000
SMTnews	1.0000	1.0000
OnWN	1.0000	1.0000

Table 3: Aggregate performance on test sets

	Pearson	Spearman
ALL	1.0000	1.0000
ALLnrm	1.0000	1.0000
Mean	1.0000	1.0000

4.3. Error analysis

In order to analyse situations in which our system most likely fails, during every testing phase we created a file containing sentence pairs for which our model prediction and human judgement mostly differ. We discovered that there are few specific kinds of problems that we didn't manage to successfully overcome.

- If there are many words shared among sentences, but without much semantic similarity in contexts of these words, our system will predict the similarity score that is much bigger than expected. This kind of sentence pairs especially appear in *MSRpar* and *SMTeuroparl* datasets. To solve this problem we should try to add more features describing semantic relationships between words, rather than their individual meanings.
- On the other hand, if the sentences are semantically almost the same, but the words used in the sentences are not equal, our system will predict the score much lower than expected. This problem is most often present in *MSRvid* dataset. The approach that would give us better results when it comes to this kind of situations could consist of removing some features that focus directly on comparing word equality and add other ones that take into account their semantic meaning.
- Next interesting category of sentence pairs that cause our system to misjudge are pairs in which one sentence starts with another one and then adds some additional information. The result of this situation is that our system predicts the similarity score severely lower than what it should be. This kind of sentences appear mostly in *MSRvid* dataset.

5. Conclusion and Future Work

In this paper we presented our system for assessing the semantic textual similarity between two short texts based on machine learning.

TODO

References

Eneko Agirre, Daniel Cer, Mona Diaba, 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.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, 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 (TIST)*, 2(3):27.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 system demonstrations*, pages 169–174. Association for Computational Linguistics.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norviq, Join Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182.

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 385–393. Association for Computational Linguistics.

SemEval. 2015. Semeval-2015 task 2: Semantic textual similarity. [Online; accessed 7-June-2015].

Wikipedia. 2015. Semantic similarity — wikipedia, the free encyclopedia. [Online; accessed 7-June-2015].