# UNBNLP at SemEval-2016 Task 1: Semantic Textual Similarity: A Unified Framework for Semantic Processing and Evaluation

**Milton King** and **Waseem Gharbieh** and **SoHyun Park** and **Paul Cook**
Faculty of Computer Science, University of New Brunswick
Fredericton, NB E3B 5A3, Canada
{milton.king,waseem.gharbieh,sohyun.park,paul.cook}@unb.ca

## Abstract

In this paper we consider several approaches to predicting semantic textual similarity using word embeddings, as well as methods for forming embeddings for larger units of text. We compare these methods to several baselines, and find that none of them outperform the baselines. We then consider both a supervised and unsupervised approach to combining these methods which achieve modest improvements over the baselines.

## 1 Introduction

Word embeddings (Mikolov et al., 2013) have recently led to improvements in a wide range of tasks in natural language processing. A number of approaches to forming embeddings for sentences, paragraphs, and documents have also recently been proposed (e.g., Le and Mikolov, 2014; Kiros et al., 2015). These methods seem particularly well suited to the task of predicting semantic textual similarity (STS), and indeed have been shown to work very well on similar tasks (Kiros et al., 2015).

This paper describes the system of UNBNLP at SemEval-2016 Task 1. We first implement several baseline approaches to STS based on cosine similarity of count-based vectors representing sentences, with a variety of approaches to term weighting. We then consider approaches drawing off of word2vec (Mikolov et al., 2013), paragraph vectors (Le and Mikolov, 2014), and skip-thoughts (Kiros et al., 2015). We find that none of these approaches improve over any of our baselines.

We then consider combining information from these individual methods to measuring STS. We consider an unsupervised approach based on the average of the predicted similarities for a number of these individual approaches. We further consider a supervised approach in which we train ridge regression with features corresponding to the similarities from these individual methods. Each of these methods for combining information achieves modest improvements over the baselines.

## 2 Measuring short text similarity

We present several methods for measuring STS in Section 2.1. We then present approaches to combining these methods in Section 2.2.

### 2.1 Individual methods

#### 2.1.1 Baselines

We present three baseline methods. In all of these baselines, each sentence in a pair of sentences is represented as a vector, where each dimension corresponds to a word type (i.e., a word form).

In the first approach, referred to as BASELINE-BIN, the dimensions hold binary values indicating whether the corresponding type occurs in the sentence. In the second approach, BASELINE-FREQ, the dimensions hold the frequency of the corresponding type in the sentence. For the third approach, BASELINE-TF-IDF, each dimension holds the tf-idf weight for the corresponding type in the sentence. Idf values were calculated over a 2015 dump of English Wikipedia from 1 September 2015, which was pre-processed using wp2txt[1] to remove markup.

[1] https://github.com/yohasebe/wp2txt



*Proceedings of SemEval-2016*, pages 1316–1319,
San Diego, California, June 16-17, 2016. 

For all baseline methods, the similarity between two sentences is calculated as the cosine between the vectors representing them. In these baseline methods, the documents are tokenized using an approach suggested by Speriosu et al. (2011) — the text is first split based on whitespace; for each token, if it contains at least one alphanumeric character, then all leading and trailing non-alphanumeric characters are stripped. Stopwords are removed based on a stopword list,[2] and case folding is applied.

#### 2.1.2 Word2vec

We considered two methods based on word embeddings from word2vec (Mikolov et al., 2013). For each sentence, we formed a vector corresponding to the element-wise summation, and product, of the word embeddings for each token in that sentence. We then measure the similarity of two sentences as the cosine between their vector representations. We refer to these methods as WORD2VEC-SUM and WORD2VEC-PROD, respectively.

For this method, we used pre-trained word2vec vectors provided by Google.[3] These vectors have 300 dimensions, and were trained on a corpus of documents from Google News that contained approximately 100 billion tokens.

For this method, sentences were tokenized by splitting on whitespace, and then removing non-alphanumeric characters. The text was also case-folded.

#### 2.1.3 Paragraph vectors

Paragraph Vectors (Le and Mikolov, 2014) is an extension of word2vec (Mikolov et al., 2013) to text of arbitrary length. In our implementation, we used the Distributed Memory Model of Paragraph Vectors (PV-DM) to represent each sentence as a vector. The similarity between two sentences was then computed as the cosine of their vector representations. We refer to this approach as PARAGRAPH-VECTORS.

The gensim[4] implementation of the PV-DM model was trained on a roughly 540 million token sample of English Wikipedia. To tokenize the Wikipedia corpus, the text was first split based on whitespace; then, all non-alphanumeric characters, except for +, -, $ and %, were removed. The remaining tokens wore case-folded. Tokens that did not have a Unicode encoding, or that occurred less than 5 times in the corpus were removed. During training, every paragraph in the corpus was treated as a separate paragraph in the model.[5]

The dimensionality of the word and paragraph representations was set to 400. A window size of 8 was used. The negative sampling parameter was set to 20. The subsampling parameter was set to $10^{-5}$. After training the model, the vector representing each sentence was inferred.

#### 2.1.4 Skip-thoughts

Skip-thoughts (Kiros et al., 2015) can be viewed as an extension of the word2vec skipgram model for obtaining vector representations of sentences. Skip-thoughts is primarily an encoder–decoder model composed of gated recurrent units (GRUs). A GRU (Cho et al., 2014) is a recurrent neural network used for sequence modeling (Chung et al., 2014). It is similar to long short-term memory (Hochreiter and Schmidhuber, 1997), but with a simplified gating architecture that does not include separate internal memory cells. The encoder receives a sequence of tokens from a sentence, and the decoder attempts to predict the sentence before the input sentence, and the sentence after it. Once the model has been trained, the vector representation of a sentence can be extracted from the learned encoder by inputting the sequence of tokens that makes up the sentence.

We used the pre-trained combine-skip model provided by Kiros et al. (2015) to build the vector representation of sentences. This produces a 4800 dimensional vector for each sentence by concatenating the vector representations from the uni-skip model and the bi-skip model. The uni-skip model is a unidirectional encoder that encodes the input tokens of a sentence in their original order, and outputs a 2400 dimensional vector. The bi-skip model is a bidirectional model that encodes the input tokens of a sentence in their original order, and in their reversed order, outputting a 1200 dimensional vector for each direction. The similarity between two sentences is

[2] http://www.lextek.com/manuals/onix/stopwords1.html

[3] https://code.google.com/archive/p/word2vec/

[4] https://radimrehurek.com/gensim/

[5] This model can be applied to various units of text, e.g., sentence, paragraph, document.

| Method | Answer–answer | Headlines | Plagiarism | Post-editing | Question–question | All |
|---|---|---|---|---|---|---|
| BASELINE-BIN | 0.50937 | 0.70636 | **0.80108** | 0.76370 | 0.61827 | 0.67881 |
| BASELINE-FREQ | 0.44204 | **0.72754** | 0.79604 | 0.79483 | **0.65749** | 0.68122 |
| BASELINE-TF-IDF (Run 1) | 0.45928 | 0.66593 | 0.75778 | 0.77204 | 0.61710 | 0.65271 |
| WORD2VEC-PROD | 0.39310 | 0.60667 | 0.71528 | 0.21306 | 0.10847 | 0.41322 |
| WORD2VEC-SUM | 0.13521 | 0.14328 | 0.23290 | -0.02673 | 0.25153 | 0.14303 |
| PARAGRAPH-VECTORS | 0.41123 | 0.69169 | 0.60488 | 0.75547 | -0.02245 | 0.50206 |
| SKIP-THOUGTS | 0.27148 | 0.23199 | 0.49643 | 0.48636 | 0.17749 | 0.33446 |
| SKIP-THOUGTS-REG | 0.28626 | 0.51019 | 0.66708 | 0.69947 | 0.40459 | 0.51299 |
| AVERAGE (Run 2) | **0.58520** | 0.69006 | 0.78923 | **0.82540** | 0.58605 | 0.69635 |
| REGRESSION (Run 3) | 0.55254 | 0.71353 | 0.79769 | 0.81291 | 0.62037 | **0.69940** |

**Table 1:** Pearson correlation for each method, on each dataset, as well as the weighted average correlation over all datasets ("All"). The best method on each dataset, and over all datasets, is shown in boldface.

then computed by taking the cosine similarity of their vector representations. This method is referred to as SKIP-THOUGTS.

We further considered a supervised approach based on skip-thought vectors. We again formed a vector representing each sentence using the pre-trained model provided by Kiros et al. Then, following Kiros et al., we represented each pair of sentences as a vector consisting of the concatenation of the componentwise product, and absolute difference, of the vectors representing the sentences. That is, if $\vec{u}$ and $\vec{v}$ are the $d$-dimensional skip-thought vectors representing two sentences, we represent this sentence pair as a $2d$-dimensional vector consisting of the concatenation of $\vec{u} \circ \vec{v}$ and $|\vec{u} - \vec{v}|$. We trained ridge regression using gold-standard STS data from 2012, 2013 and 2015, and then used this model to predict similarity for the test sentence pairs. We refer to this model as SKIP-THOUGTS-REG. We implemented this model after submitting our official runs.

### 2.2 Method combinations

We used two different methods — one unsupervised, and one supervised — to combine the individual methods in an effort to develop a stronger system.

For the unsupervised method, AVERAGE, we computed the average of BASELINE-BIN, BASELINE-TF-IDF, WORD2VEC-PROD, PARAGRAPH-VECTORS, and SKIP-THOUGTS. We did not consider BASELINE-FREQ here because it is quite similar to BASELINE-BIN, which performed better on development data.

For the supervised approach to combining individual methods, we trained ridge regression over the similarities produced by the following methods: BASELINE-BIN, BASELINE-TF-IDF, PARAGRAPH-VECTORS, and SKIP-THOUGTS. The ridge regression was trained using the gold standard data provided for STS tasks in 2012, 2013, and 2015; this model was then used to predict similarities for sentence pairs in the test data. We refer to this method as REGRESSION.

## 3 Results

Results for each method, on each dataset, are shown in Table 1. We first consider the baseline approaches. On development data from previous STS tasks, BASELINE-TF-IDF gave higher correlations than baselines based on word presence (BASELINE-BIN) or word frequency (BASELINE-FREQ). Moreover, this was a challenging baseline to beat, and was among the best methods we considered on the development data. It was therefore submitted as one of our official runs. However, on the test data, BASELINE-TF-IDF had the lowest average correlation of the three baseline approaches considered.

In terms of the methods based on word2vec, representing a sentence as the componentwise product of the vectors for the words in that sentence (WORD2VEC-PROD) performed much better than the approach based on vector addition (WORD2VEC-SUM). PARAGRAPH-VECTORS outperformed both word2vec approaches. However, none of these word embedding-based methods performed as well as any of the baselines.

Naively measuring similarity as the cosine between skip-thought vectors for the sentences in a pair (SKIP-THOUGTS) led to relatively poor perfor-

mance. Training ridge regression based on features derived from skip-thought vectors (SKIP-THOUGTS-REG, described in Section 2.1.4) led to substantial improvements, although again this approach did not beat any of the baselines.

AVERAGE and REGRESSION — both submitted as official runs — combine several of the individual methods together, and both achieve correlations that are, overall, better than those of any of the baselines. However, the improvements are relatively modest. Although there is some variation across the datasets, AVERAGE and REGRESSION perform very similarly overall. AVERAGE, however, has an advantage, in that it is an unsupervised approach.

## 4 Conclusions

In this paper we first considered several baseline approaches to STS. We then considered approaches based on word2vec, paragraph vectors, and skip-thoughts. We found that none of these approaches improved over any of the baselines. We further considered combining these approaches via averaging, and a supervised approach based on regression, and achieved modest improvements over the baselines.

## Acknowledgments

This work is financially supported by the Natural Sciences and Engineering Research Council of Canada, the New Brunswick Innovation Foundation, and the University of New Brunswick.

## References

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar, pages 103–111.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of Deep Learning and Representation Learning Workshop: NIPS 2014*. Montreal, Canada.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 3276–3284.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the First workshop on Unsupervised Learning in NLP*. Edinburgh, Scotland, pages 53–63.