HHU at SemEval-2016 Task 1: Multiple Approaches to Measuring Semantic Textual Similarity

Matthias Liebeck, Philipp Pollack, Pashutan Modaresi, Stefan Conrad
Institute of Computer Science
Heinrich Heine University Düsseldorf
D-40225 Düsseldorf, Germany
{liebeck,modaresi,conrad}@cs.uni-duesseldorf.de
philipp.pollack@hhu.de

Abstract

This paper describes our participation in the SemEval-2016 Task 1: *Semantic Textual Similarity* (STS). We developed three methods for the English subtask (STS Core). The first method is unsupervised and uses WordNet and word2vec to measure a token-based overlap. In our second approach, we train a neural network on two features. The third method uses word2vec and LDA with regression splines.

1 Introduction

Measuring semantic textual similarity (STS) is the task of determining the similarity between two different text passages. The task is important for various natural language processing tasks like topic detection or automated text summarization because languages are versatile and authors can express similar content or even the same content with different words. Predicting semantic textual similarity has been a recurring task in SemEval challenges (Agirre et al., 2015; Agirre et al., 2014; Agirre et al., 2013; Agirre et al., 2012). As in previous years, the purpose of the STS task is the development of systems that automatically predict the semantic similarity of two sentences in the continuous interval $[0, 5]$ where 0 represents a complete dissimilarity and 5 denotes a complete semantic equivalence between the sentences (Agirre et al., 2015).

The organizers provide sentence pairs whose semantic similarities have to be predicted by the contestants. The quality of a system is determined by calculating the Pearson correlation between the predicted values and a human gold standard that has been created by crowdsourcing. The data from previous STS tasks can be used for training supervised methods.

The test data consists of text content from different sources. In this year's shared task, the systems are tested on five different categories with different topics and varying textual characteristics like text length or spelling errors: *answer-answer*, *plagiarism*, *postediting*, *headlines*, and *question-question*

The remainder of the paper is structured as follows: Section 2 discusses related approaches to automatically determining semantic textual similarity. Section 3 describes our three methods in detail. We discuss their results in section 4. Finally, we conclude in chapter 5 and outline future work.

2 Related Work

In the last shared tasks, most of the teams used natural languages processing techniques like tokenization, part-of-speech tagging, lemmatization, named entity recognition and word embeddings. External resources like WordNet (Miller, 1995) and word2vec (Mikolov et al., 2013) are commonly used. In (Agirre et al., 2012) and (Agirre et al., 2013), the organizers provide a list and a comparison of the tools and resources used by the participants in the first two years, respectively.

In each year, the organizers provide a baseline value by calculating the cosine similarity of the binary bag-of-words vectors from both sentences in each sample. Since 2013, TakeLab (Šarić et al., 2012), the best ranked system in 2012, has also been used as another baseline value.

Most of the teams used machine learning in 2015

Proceedings of SemEval-2016, pages 607–613,
San Diego, California, June 16-17, 2016. ©2016 Association for Computational Linguistics

(Agirre et al., 2015). In 2014, the best two submitted runs were from unsupervised systems.

The work most closely related to our Overlap method is (Han et al., 2015), which uses a two-phased approach called *Align-and-Differentiate*. In the first phase, they compute an alignment score. Afterwards, they modify the alignment score in a differentiate phase by subtracting a penalty score for terms that can not be aligned. The idea behind the computation of our alignment scores is the same: For each sample, we average over the crosswise similarities between the sentences by aligning them, accumulating similarities between tokens and dividing by sentence lengths. The results of the alignment score in our Overlap method differ because (i) our alignment is different, (ii) we use another similarity function for tokens, and (iii) our preprocessing is different.

In (Vu et al., 2015), the similarity between LDA vectors calculated from documents is used together with syntactic and lexical similarity measures to compute the similarity between text fragments. This idea is also incorporated in our Deep LDA method. Moreover, both approaches use different flavors of regression analysis for the final model prediction. Regression analysis was also used in (Sultan et al., 2015), where the authors combine an unsupervised method with ridge regression analysis. Our approach differs in the sense that it introduces k-nearest neighbors as a lazy training layer before the regression analysis phase to decrease the effect of noisy data points.

3 Methods

In this section, we describe our three system runs. The ideas behind our methods are independent of the word order in a sentence. Our first method is unsupervised, whereas the other two methods are supervised. The first and second method share the same preprocessing.

3.1 Run 1: Overlap Method

Our first method is unsupervised. It measures the overlap between the tokens in sentence s_1 and the tokens in sentence s_2.

3.1.1 Preprocessing

For preprocessing the input text, we first process each sentence with Stanford CoreNLP (Manning et al., 2014). Afterwards, we use Hunspell[1] with the latest OpenOffice English dictionaries to suggest spelling corrections for tokens with at least two characters in length. For each token, we calculate the Levenshtein distance for all suggestions. If suggestions have the same lowest distance, we choose the longest word and replace the former misspelt word. Abbreviations are also replaced by their full forms. Afterwards, we process the corrected sentence with Stanford CoreNLP again. We use the WordnetStemmer from the *Java Wordnet Interface* (Finlayson, 2014) to look up lemmas with the help of WordNet (Miller, 1995). If the WordnetStemmer can not provide a lemma for a token, we use the predicted lemma from the Stanford CoreNLP.

Instead of accessing all tokens in a sentence, we start from the root token and recursively follow outgoing dependency edges and add all visited tokens to a list. This approach improves our results slightly because some tokens will be ignored. Furthermore, the tokens are filtered for stopwords[2].

3.1.2 Method

The Overlap method measures the token-based overlap between two sentences. Therefore, we need to define a similarity function for tokens: We first try to identify a textual similarity of 1 by comparing the lower case lemmas of both tokens or by checking if their most common WordNet synsets are the same. We assess their similarity as 0.5 if they share any synset. If this is not the case, we use word2vec (Mikolov et al., 2013) with the 300-dimensional *GoogleNews-vectors-negative300* model. We look up both words (or their lemmas if the words are not present in the model) and calculate the cosine similarity of their word embeddings. Otherwise, we return a default value.

This yields the following similarity function for two tokens:

[1] http://hunspell.github.io/

[2] http://xpo6.com/list-of-english-stop-words/

$$\mathrm{sim}(t_1, t_2) := \begin{cases} 1 & \text{if } t_1.\text{lemma} == t_2.\text{lemma} \\ 1 & \text{if } t_1 \text{ and } t_2 \text{ have the same most common synset} \\ 0.5 & \text{if } t_1 \text{ and } t_2 \text{ share any synset} \\ d(t_1, t_2) & \text{if } t_1 \text{ and } t_2 \text{ have word embeddings} \\ \textit{default} & \text{otherwise} \end{cases}$$

where $d(t_1, t_2)$ denotes the cosine similarity between the two word embeddings of the tokens.

Given a token t from one sentence, we calculate its similarity to another sentence S by taking the maximum similarity between t and all tokens of S:

$$\mathrm{msim}(t, S) := \max_{t_2 \in S} \mathrm{sim}(t, t_2)$$

We define the similarity score between two sentences in $[0, 1]$ as follows:

$$\mathrm{ssim}(s_1, s_2) := \frac{\sum\limits_{t \in s_1} \mathrm{msim}(t, s_2)}{2 \cdot |s_1|} + \frac{\sum\limits_{t \in s_2} \mathrm{msim}(t, s_1)}{2 \cdot |s_2|}$$

To predict the semantic similarity score in $[0, 5]$, we multiply ssim by 5, however, this does not change our evaluation results because the Pearson correlation is scale invariant:

$$\mathrm{STS}(s_1, s_2) := 5 \cdot \mathrm{ssim}(s_1, s_2)$$

We observed that some samples in the STS 2016 test data consist almost entirely of stopwords. For example, the STS 2016 evaluation data contained a sample with the sentences "*I think you should do both.*" and "*You should do both.*" before the final filtering. After filtering stop words, the first sentence would only contain the word "*think*" and the second sentence would be empty, which would result in a predicted score of zero. To avoid these extreme cases, we do not filter stop words if this would result in a sentence length of less than two tokens in both sentences.

3.2 Run 2: Same Word Neural Network Method

We train a neural network with 3 layers and a sigmoid activation function in Accord.NET (de Souza, 2014). Our network consists of 2 neurons in the input layer, 3 neurons in the hidden layer and 1 neuron in the output layer, as illustrated in Figure 1. The layer weights are initialized by the Nguyen-Widrow function (Nguyen and Widrow, 1990). We use the Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963) to train our network on the STS Core test data from 2015 and 2014.

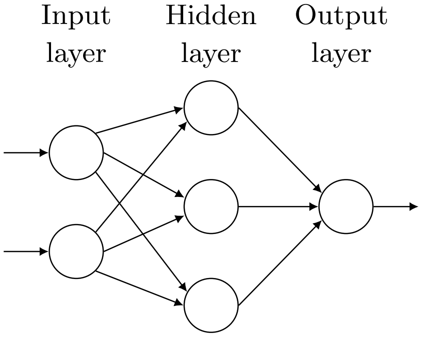

Figure 1: Architecture of our neural network

All samples are preprocessed as described in section 3.1.1. For each sample (s_1, s_2, gs) in the training set, we create a vocabulary list of the lowercase lemmas from both sentences. Lemmas that share a most common synset in WordNet are grouped together. Let n be the size of the vocabulary. We create two bag-of-words vectors bow_{s_1} and bow_{s_2}. For each lemma l, we calculate the minimum number of times l occurs in each sentence and the delta between the minimum and the maximum:

$$\begin{aligned} \min_i &:= \min(\mathrm{bow}_{s_1}[i], \mathrm{bow}_{s_2}[i]) \\ |\Delta_i| &:= |\mathrm{bow}_{s_1}[i] - \mathrm{bow}_{s_2}[i]| \end{aligned}$$

As input vectors for the neural net, we build two sums per sample and use them as the two dimensional feature vector $(\mathrm{sameWords}, \mathrm{notSameWords})$ for the expected output gs:

$$\begin{aligned} \mathrm{sameWords} &:= \sum_{i=1}^{n} \min_i \\ \mathrm{notSameWords} &:= \sum_{i=1}^{n} |\Delta_i| \end{aligned}$$

Table 1 shows an example of the same word neural network method for the two input sentences "*Tim plays the guitar*" and "*Tim likes guitar songs*", which have the input vector $(2, 3)$.

i	Lemma	bow_{s_1}	bow_{s_2}	min_i	$\lvert\Delta_i\rvert$
1	tim	1	1	1	0
2	play	1	0	0	1
3	guitar	1	1	1	0
4	like	0	1	0	1
5	song	0	1	0	1
$\sum$				2	3

Table 1: An example for creating the two-dimensional feature vector for the *Same Word Neural Network* method

We trained the neural net until the error rate between two iterations did not change more than $\varepsilon = 10^{-5}$.

3.3 Run 3: Deep LDA Method

We represent the semantic similarity between two documents s_1 and s_2 by means of a vector $F = [f_1, f_2, f_3, f_4] \in \mathbb{R}^4$, where each component of F is responsible for modelling a different aspect of the semantic similarity, namely the *surface-level similarity*, *context similarity*, and the *topical similarity*.

Surface-level Similarity

The surface-level similarity can to some extent (although not entirely) capture the semantic similarity between documents. Let s_1 and s_2 be the reference and the candidate documents respectively. We compute the components $f_1, f_2 \in \mathbb{R}$ as follows:

$$f_1(s_1, s_2, N) = \frac{m_N}{l_N^{s_1}}$$

$$f_2(s_1, s_2, N) = \left(\prod_{n=1}^{N} \frac{m_N}{l_n^{s_2}} \right)^{\frac{1}{N}}$$

where m_N is the number of matched N-grams between s_1 and s_2, $l_N^{s_1}$ denotes the total number of N-grams in s_1 and $l_n^{s_2}$ is the total number of n-grams in s_2. f_1 is the common ROUGE (Lin, 2004) metric used in automatic text summarization and f_2 is a modified version of the BLEU (Papineni et al., 2002) metric (standard machine translation metric) where the brevity penalty is eliminated. Note that f_1 can be interpreted as the recall-oriented surface similarity and f_2 as the precision-oriented one.

Context Similarity

In order to model the context similarity between documents, we use word embeddings that learn semantically meaningful representations for words from local co-occurrences in sentences. More specifically we use *word2vec* (Mikolov et al., 2013) which seems to be a reasonable choice to model context similarity as the word vectors are trained to maximize the log probability of context words. We denote the context similarity of two documents s_1 and s_2 by $f_3 \in \mathbb{R}$ and compute it as follows:

$$\begin{aligned} f_3(s_1, s_2) &= \cos(\tilde{v}_{s_1}, \tilde{v}_{s_2}) \\ &= \cos\left(\frac{\sum_{v \in s_1} v}{|s_1|}, \frac{\sum_{v' \in s_2} v'}{|s_2|} \right) \end{aligned}$$

where v is the dense vector representation of a token and $\tilde{v}$ represents the centroid of the word vectors in a document.

Topical Similarity

To model the topical similarity between two documents, we use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to train models on the English Wikipedia. For both documents s_1 and s_2, we compute the topic distributions θ_1 and θ_2 and use the Hellinger distance to measure the similarity between the documents. This can be formally written as

$$f_4(s_1, s_2) = 1 - \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} \left(\sqrt{\theta_{1_i}} - \sqrt{\theta_{2_i}} \right)^2}$$

where k represents the number of learned LDA topics.

Similarity Prediction

In order to predict the semantic similarity between two documents, we use a combination of k-NN and Multivariate Adaptive Regression Splines (MARS) (Friedman, 1991).

Let $T = \{(s_1, s_1', gs_1), \ldots, (s_m, s_m', gs_m)\}$ be the training set consisting of m document pairs together with their corresponding gold standard semantic similarity and $(s_i, s_i') \notin T$ be a document pair for which the semantic similarity has to be computed. We construct a set $\mathcal{F} = \{(F_1, gs_1), \ldots, (F_m, gs_m)\}$ where each F_j is the four-dimensional vector representation of the semantic similarity between s_j and s_j'. Moreover, we

Sentence 1	Sentence 2	gs	STS
Unfortunately the answer to your question is we simply do not know.	Sorry, I don't know the answer to your question.	4	4.05800
You should do it.	You can do it, too.	1	4.39817
Unfortunately the answer to your question is we simply do not know.	My answer to your question is "Probably Not".	1	3.70982
$P(A\|B)$ is the conditional probability of A, given B.	$P(B\|A)$ is the conditional probability of B given A.	3	4.32017

Table 2: Examples for the results of the Overlap method with the corresponding gold standards

compute the vector F_i. Next, we construct a set $\mathcal{F}_k$ containing the k-nearest neighbors to the vector F_i. In order to calculate the distances between the vectors, we use the Euclidean distance. Finally, we construct a vector gs_k containing the gold standard similarity values of the k-nearest neighbors and feed it into a MARS model to predict the semantic similarity of the pair (s_i, s_i'). The choice of MARS is due to its capability to automatically model non-linearities between variables.

4 Results

We report the results of our three approaches for the STS Core test from 2016 and 2015.

4.1 STS 2016 Results

In this years shared task, 117 runs were submitted. We achieved weighted mean Pearson correlations of 0.71134, 0.67502 and 0.62078. In this year's run, our best result was the Overlap method, followed by the Same Word Neural Network method and the Deep LDA approach. Table 2 shows examples of good and bad results of our Overlap method on the 2016 data. Detailed results of our runs are given in Table 3 per test set. Our three approaches achieved different results.

From a semantic point of view, the most obvious value for the default value in our Overlap method is 0. However, we have discovered that a default value 0.15 returned better results on the STS Core test data from 2015 and also chose this default value for our submission.

In the Deep LDA approach, we set the parameter $N = 2$, although the use of unigrams did not show any significant statistical difference in the results. We choose the number of topics in the LDA model to be 300. In the prediction phase of the algorithm, we select $k = 100$ nearest neighbors from the data sets provided from 2012 to 2015.

Data set	Run 1	Run 2	Run 3
answer-answer	0.50435	0.42673	0.47211
headlines	0.77406	0.75536	0.58821
plagiarism	0.83049	0.79964	0.62503
postediting	0.83846	0.84514	0.84743
question-question	0.60867	0.54533	0.57099
Weighted Mean	0.71134	0.67502	0.62078

Table 3: Pearson correlation of the 2016 test data

Data set	Run 1	Run 2	Run 3
answers-forums	0.74163	0.70387	0.79987
answers-students	0.73685	0.76658	0.76733
belief	0.74046	0.73319	0.78242
images	0.82032	0.80813	0.84747
headlines	0.75358	0.74363	0.76076
Weighted Mean	0.76295	0.75922	0.79168

Table 4: Pearson correlation of the 2015 test data

4.2 STS 2015 Results

We list the results of our methods for the 2015 test data in Table 3 to discuss the effect of different evaluation sets. It is interesting to see that the Deep LDA method performed best out of our three systems on 2015. Its results on 2016 were surprisingly lower. We attribute this difference to the lack of domain specific training data for 2016. As an unsupervised approach, the Overlap method has fewer problems with the domain change.

It should be noted that the gold standard of the 2015 test data was available during the development of our methods. For the training phase, the Same Word Neural Network method used the STS Core test from 2014. The Deep LDA method was trained on the data from 2012 to 2014.

5 Conclusion and Future Work

We have presented three approaches to measure textual semantic similarity. This year, our unsupervised method achieved the best result. By comparing our result for 2016 and 2015, we showed that the approaches yielded different results in a different order.

In our future work, we will try to modify the Overlap method, by also using a penalty score and by applying certain similarity score shifters, for instance modifying the score by applying a date extraction with a specific distance function for dates. We tried to group words into phrases by using a sliding window approach with a shrinking window size and matching phrases in word2vec. In our initial attempt, this worsened the results for the Overlap method. We will adjust the similarity function to increase the weight of phrases in comparison to unigrams.

We aim to adapt the techniques for German and Spanish.

Acknowledgments

This work was partially funded by the PhD program *Online Participation*, supported by the North Rhine-Westphalian funding scheme *Fortschrittskollegs* and by the German Federal Ministry of Economics and Technology under the ZIM program (Grant No. KF2846504). The authors want to thank the anonymous reviewers for their suggestions and comments.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In **SEM 2012: 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 (SemEval 2012)*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91. Association for Computational Linguistics and Dublin City University.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

César Roberto de Souza. 2014. The Accord.NET Framework. `http://accord-framework.net`.

Mark Finlayson. 2014. Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation. In *Proceedings of the Seventh Global Wordnet Conference*, pages 78–85.

Jerome H. Friedman. 1991. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67.

Lushan Han, Justin Martineau, Doreen Cheng, and Christopher Thomas. 2015. Samsung: Align-and-Differentiate Approach to Semantic Textual Similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 172–177. Association for Computational Linguistics.

Kenneth Levenberg. 1944. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 74–81.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Donald W. Marquardt. 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop*.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Derrick Nguyen and Bernard Widrow. 1990. Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 21–26. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318. Association for Computational Linguistics.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 148–153. Association for Computational Linguistics.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. TakeLab: Systems for Measuring Semantic Text Similarity. In **SEM 2012: 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 (SemEval 2012)*, pages 441–448. Association for Computational Linguistics.

Tu Thanh Vu, Quan Hung Tran, and Son Bao Pham. 2015. TATO: Leveraging on Multiple Strategies for Semantic Textual Similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 190–195. Association for Computational Linguistics.