# UWB at SemEval-2016 Task 2: Interpretable Semantic Textual Similarity with Distributional Semantics for Chunks

**Miloslav Konopík** and **Ondřej Pražák** and **David Steinberger** and **Tomáš Brychcín**
NTIS – New Technologies for the Information Society,
Department of Computer Science and Engineering,
Faculty of Applied Sciences, University of West Bohemia, Technická 8, 306 14 Plzeň
Czech Republic
{konopik,brychcin}@ntis.zcu.cz,
{ondfa,fenic}@kiv.zcu.cz

## Abstract

We introduce a system focused on solving SemEval 2016 Task 2 – Interpretable Semantic Textual Similarity. The system explores machine learning and rule-based approaches to the task. We focus on machine learning and experiment with a wide variety of machine learning algorithms as well as with several types of features. The core of our system consists in exploiting distributional semantics to compare similarity of sentence chunks. The system won the competition in 2016 in the "Gold standard chunk scenario". We have not participated in the "System chunk scenario".

## 1 Introduction

The goal of the *Interpretable Semantic Textual Similarity* task is to go deeper with the assessment of semantic textual similarity of sentence pairs. It is requested to add an explanatory layer that offers a deeper insight into the sentence similarities. The sentences are split into chunks and the first goal is to find corresponding chunks (with respect to their meanings) among the compared sentences. When the corresponding chunks are known, the chunks are annotated with their similarity scores and their relation types (e.g. equivalent, more specific, etc).

The task follows a pilot task from the preceding SemEval 2015 competition (Agirre et al., 2015). The best performing systems adopted various approaches, (Banjade et al., 2015) relied on handcrafter rules, (Karumuri et al., 2015) employed a classifier for relation types and they associated each relation with a precomputed similarity score and (Hänig et al., 2015) extended their word alignment algorithm for the task.

### 1.1 Math notation

The data consist of sentence pairs $\mathbf{S}_i^a$ and $\mathbf{S}_i^b$, where $a$ denotes the first item of the pair, $b$ denotes the second item of the pair and $i$ indexes the sentences (for simiplicity we further omit $i$ for sentences). We perceive a sentence $\mathbf{S}^a$ to be an ordered set of chunks $\mathbf{CH}_j^a \in \mathbf{S}^a$ and the chunks to be ordered sets of words $w_k \in \mathbf{CH}_j^a$ (and analogically for sentence $\mathbf{S}^b$).

Next we define two functions: $\texttt{sim}(\mathbf{CH}_i^a, \mathbf{CH}_j^b) \in \{0, 1, 2, 3, 4, 5\}$ for chunk similarity and $\texttt{rel}(\mathbf{CH}_i^a, \mathbf{CH}_j^b) \in \mathbf{TYPE}$ for chunk relation type.

The possible types are: $\mathbf{TYPE} = \{$*EQUI*, *OPPO*, *SPE1*, *SPE2*, *SIMI*, *REL*$\}$. These are the main types. All these types can have two modifiers (*FACT*, *POL*). The modifiers are optionally attached to the main types. For example, you can generate *SPE1_FACT*. For more information, please see the annotation guidelines[1].

## 2 System Overview

### 2.1 Preprocessing

As a first step of our approach we perform the following text preprocessing:

- *Stopwords removal* – we mark the words found in a predefined list of 32 stopwords.

[1] http://alt.qcri.org/semeval2016/task2/data/uploads/annotationguidelinesinterpretablests2016v2.2.pdf



*Proceedings of SemEval-2016*, pages 720–725,
San Diego, California, June 16-17, 2016. ©2016 Association for Computational Linguistics

- *Special character removal* we remove special characters that violate the tokenization. E.g. in one of the datasets, dots, commas, quotation marks and other punctuation characters were present in tokens.

- *Lowercasing* – we remove casing from the words.

- *Lemmatization* – we find lemmas with the Stanford CoreNLP tool (Manning et al., 2014).

Our preprocessing rather adds new information and does not modify the original information. Thus, the original word and all the generated variants are always available. In this way, we can generate the output file with identical words (including the special characters) from the input. The dataset are already tokenized.

### 2.2 Chunk Semantic Similarity

The core of our system is based upon computing semantic similarity of sentence chunks. More precisely, we are looking for the best estimation of $\texttt{sim}(\mathbf{CH}_i^a, \mathbf{CH}_j^b)$. The sim score should describe semantic similarity of a given chunk pair – the higher score the more easily both chunks can be replaced with each other without chaining the meaning of both sentences. The similarity score ranges from 0 to 5, where 0 is the lowest similarity and 5 is the highest similarity. Eg. the $\texttt{sim}(\text{“a new laptop”}, \text{“a new notebook”}) = 5$ and $\texttt{sim}(\text{“a new laptop”}, \text{“an old rock”}) = 0$. We use the chunk similarity as a feature in our machine learning approach (Section 3) and as a metric in our unsupervised approach (Section 4).

Our attempts to estimate the sim function are based upon estimating semantic similarity of individual words and compiling them into one number for a given chunk pair. We experiment with Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) for estimating similarity of words. We compile all the word similarities in one number that reflects semantic similarity of whole chunks via the following methods: 1) the *vector composition* method and 2) an adapted method for constructing vectors called *lexical semantic vectors*.

**Vector composition** requires that the semantics of words is described by vectors. E.g. we have vectors for all words $\vec{m}_i : \forall w_i \in \mathbf{CH}_j^a$ and $\vec{n}_i : \forall w_i \in \mathbf{CH}_k^b$ in two given chunks $\mathbf{CH}_j^a$ and $\mathbf{CH}_k^b$. The vectors for words in each chunk are summed (or averaged) to obtain one vector for each chunk: $\vec{m} = \sum_i(\vec{m}_i)$ and $\vec{n} = \sum_j(\vec{n}_j)$. The vectors are then compared with cosine distance: $\texttt{sim}(\mathbf{CH}_j^a, \mathbf{CH}_k^b) = \cos(\theta) = \frac{\vec{m}\cdot\vec{n}}{|\vec{m}||\vec{n}|}$.

**Lexical semantic vectors** were originally introduced in (Li et al., 2006). We have made two modifications. We do not weight words with their information content and we use methods for distributional semantics (Word2Vec and GloVe) rather than semantic networks. The modified method is explained here. First of all, we create a combined vocabulary of all unique words from chunks $\mathbf{CH}_k^a$ and $\mathbf{CH}_l^b$: $\mathbf{L} = \texttt{unique}(\mathbf{CH}_k^a \cup \mathbf{CH}_l^b)$. Then we take all words from vocabulary $\mathbf{L}$: $w_i \in \mathbf{L}$ and look for maximal similarities with words from chunks $a$ and $b$, respectively. This way we get vectors $\vec{m}$ and $\vec{n}$ containing maximal similarities of chunk words and words from the combined vocabulary:

$$\begin{aligned} m_i &= \max_{j:1\le j\le|\mathbf{CH}_k^a|} \texttt{sim}(w_i, w_j) : \quad \forall w_i \in \mathbf{L} \\ n_i &= \max_{j:1\le j\le|\mathbf{CH}_l^b|} \texttt{sim}(w_i, w_j) : \quad \forall w_i \in \mathbf{L} \end{aligned} \tag{1}$$

where $m_i$ and $n_i$ are elements of vectors $\vec{m}$ and $\vec{n}$.

In order to obtain similarity of a chunk pair we compare their respective vectors with the cosine similarity similarly to the previous approach. The principle of the method is illustrated by the example in figure 1.

**iDF weighting.** We assume that some words are more important than others. In order to reflect this assumption, we try to weight the vectors with iDF weighting. We compute the iDF weights on the articles from English wikipedia text data (Wikipedia dump from March 7, 2015).

## 3 Machine Learning Approach

The main effort of our team was focused on the machine learning approach to the task. We divided the task into to three classification / regression tasks:

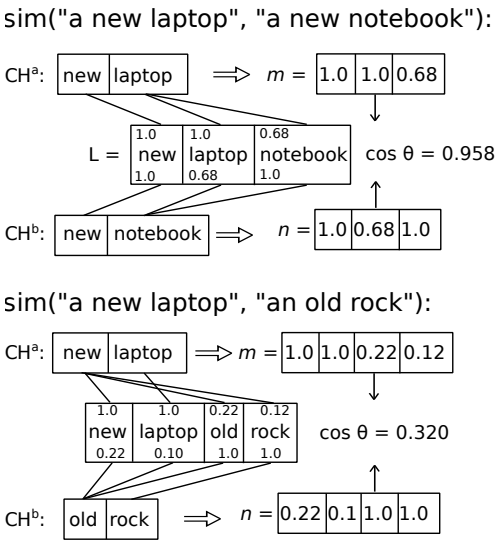


**Figure 1:** An example of the modified lexical semantic vectors method.

- *Alignment binary classification* – we decide whether two given chunks should be aligned with each other.
- *Score classification / regression* – we experiment with both classification and regression of the chunks similarity score.
- *Type classification* – we classify all aligned pairs of chunks into a predefined set of types – see Section 1.1.

### 3.1 Classifiers

We experiment with the following classifiers: *Maximum Entropy Classifier* (Berger et al., 1996), *Support Vector Machines Classifier* (Cortes and Vapnik, 1995), *Multilayer perceptron* and *Voted perceptron neural networks* (Freund and Schapire, 1999) and with *Decision / regression tree learning* (Breiman et al., 1984).

We employ the following two frameworks: *Brainy* (Konkol, 2014) and *Weka* (Hall et al., 2009).

### 3.2 Features

We divide the employed features into four categories: lexical, syntactic, semantic, external.

**Lexical features** consist of the following features: word base form overlap, word lemma overlap, chunk length difference, word sentence positions difference.

**Syntactic features** contains closest common parent comparison (we compute the closest common parent of all words for each chunk in the parse tree and retrieve the name of the parent node), parse tree path comparison (we compute the path from the root of the sentence to the chunk). POS (Part Of Speech) count difference (e.g. differences in counts of nouns, adjectives, verbs, etc). POS tagging and syntactic parsing are performed with Stanford CoreNLP (Manning et al., 2014).

**Semantic features** are described in Section 2.2. Additionally, some members of our team participated in the STS task (task 1) of the SemEval 2016 (Brychcín and Svoboda, 2016) and they annotated the semantic similarity of the whole sentences with their system for us. This score is used as one feature.

**External features** consist of the WordNet – Lin similarity metric (Lin, 1998) and the paraphrase database (Ganitkevitch et al., 2013) feature.

### 3.3 Post-processing

The alignment of chunks is generated by the binary classification of all possible chunk pairs. If one chunk is aligned with multiple chunks in the other sentence, these chunks should be merged into one chunk. Also, impossible multiple chunks to multiple chunks alignments are generated in some cases (e.g. two chunks from the first sentence belong a chunk in the second sentence but one of the two chunks from the first sentence belong also to a different chunk in the second sentence). These cases are resolved with few hand crafted rules.

## 4 Rule-based Approach

We attempt to solve the task with a rule-based approach as well. First, we define the similarity of chunks as described in Section 2.2. The similarity is then used for the chunk alignment. We employ an algorithm inspired by the IBM word model II for machine translation (Brown et al., 1993). We iterate over all chunks from sentence $\mathbf{S}_a$ and find the chunk with maximal similarity from sentence $\mathbf{S}_b$. More chunks from sentence $\mathbf{S}_a$ can be aligned to one chunk in the sentence $\mathbf{S}_b$. In this way, we obtain N:1 mapping. Then, we do the same with the reversed order of sentences and get the 1:M mapping. Then,

we compare the mappings and take the one with the highest overall similarity. In this way, it is ensured that we generate only valid mappings (unlike in the previous case of machine learning – see Section 3.3).

The relation types are then determined by an extremely simple algorithm:

- If the similarity is 5, then the relation type is *EQUI*.
- If the similarity is 4 or 3 and chunks contain the same amount of words, then the relation is *SIMI*.
- If the similarity is 4 or 3, then chunk with more words is more specific.
- If the similarity is 2 or 1, then the relation is *SIMI*.
- If the similarity is 0, then the relation type is *NOALI*.

## 5 Results

### 5.1 Experimental setup

**Machine learning approach** We employ the following classifiers and classification frameworks:

- *Alignment binary classification* – Voted perceptron (Weka).
- *Score classification* – Maximum entropy (Brainy).
- *Type classification* – Support vector machines (Brainy).

These classifiers perform best on the evaluation datasets.

We achieved the best results for estimating chunk similarity with Word2Vec and the modified lexical semantic vectors – see Section 2.2.

We experimented with reduced feature set (word overlap, word positions difference, POS tags difference, semantic similarity, global semantic similarity, paraphrase database) – run 1 and with all features – run 3. The run 1 contains the optimal combination of features. Since this combination is established on evaluation datasets it does not need to be optimal for the test datasets. To increase our chances in the completion, we also run the system with all features – run 3.

We use the provided annotated evaluation dataset (*Images*, *Headlines*, *Answer students*) for training the models. We train three models, each for one dataset. For development, we use the 10-fold cross-validation. For final test runs, we train the three models on evaluation datasets and run the system on the corresponding test datasets (e.g. *Images* evaluation dataset based model is used to annotate *Images* test data). We do not neither modify the original datasets nor annotate any additional data.

**Rule-based approach** There are little options in this approach. Again, we have achieved the best results for estimating chunk similarity with Word2Vec and the modified lexical semantic vectors – see Section 2.2. We set the threshold for the similarity score to 2.5. All lower values are set to 0. This is the run 2.

**Individual setting for different dataset** We restrained from setting individual configurations for different datasets. The setup is completely identical for all datasets.

### 5.2 Results

In this section, we summarize the official results for the SemEval 2016 competition – see table 1. The results are calculated for the following dataset: *Headlines*, *Images* and *Answer students*. The results show F1 scores for chunk alignment (*Ali*), determination of the relation type (*Type*), chunk similarity score (*Score*) and combination of relation type and score similarity (*T+S*). The bold numbers are the overall best scores. We participated only in the gold standard chunk scenario.

The results clearly show that the unsupervised run 2 perform much worse than the supervised runs 1 and 3. We expected that. However, it is worth of noticing that the unsupervised alignment algorithm inspired by machine translation alignment placed quite well. In fact, it is newer looses more than 3% from the best alignment score in all datasets. The overall rank of the run 2 places in the top half among all system with exception of the *answer student* dataset. The poor performance of the run 2 on this dataset is most likely caused by the fact that the hand-crafted rules were prepared for the *images* and

| Run | Ali | Type | Score | T+S | Rank |
|---|---|---|---|---|---|
| | Images dataset (756 sentence pairs) | | | | |
| 3 | 0.8922 | **0.6867** | 0.8408 | **0.6708** | 1 |
| 1 | **0.8937** | 0.6829 | 0.8397 | 0.6672 | 2 |
| 2 | 0.8713 | 0.6346 | 0.8083 | 0.6206 | 6 |
| | Headlines dataset (750 sentence pairs) | | | | |
| 3 | 0.8987 | 0.6412 | **0.8382** | 0.6296 | 6 |
| 1 | 0.8979 | 0.6319 | 0.8346 | 0.6212 | 7 |
| 2 | 0.8897 | 0.6146 | 0.815 | 0.6013 | 8 |
| | Answer students dataset (330 sentence pairs) | | | | |
| 1 | 0.8644 | 0.6299 | 0.8089 | 0.6248 | 3 |
| 3 | 0.8588 | 0.6167 | 0.8038 | 0.6114 | 5 |
| 2 | 0.8752 | 0.4806 | 0.7826 | 0.4748 | 17 |
| | **Overall results** (mean) | | | | |
| 1 | 0.6672 | 0.6212 | 0.6248 | 0.6377 | 1 |
| 3 | **0.6708** | 0.6296 | 0.6114 | 0.6373 | 2 |
| 2 | 0.6206 | 0.6013 | 0.4748 | 0.5656 | 12 |

**Table 1:** Official system evaluation.

*headlines* datasets and they are clearly not applicable on the *answer student* which is substantially different.

The runs 1 and 3 perform very similarly. The optimized feature set of the run 1 helps especially in the *answer student* dataset. However, the differences between these runs are too small and they can be caused by chance. It is worth of noticing that the run 1 is not the best one in any of the datasets and it still wins in the overall results table. The reason is that it provides the most consistent results among all other runs of all systems in the competition.

In order to provide additional information about the features effectiveness, we have evaluated them on the final test datasets. In many cases, the obtained results are not conclusive. On some datasets, the features help slightly on others they even decrease the performance. However, the following three features have significant influence on the final results: modified lexical semantic vectors (+3% of the mean of T+S F1 scores), shared words (+2%), POS tags difference (+2%). The modified lexical semantic vectors method performed better than vector composition by 1% for the machine learning approach and by 2% for the rule-based approach in average. By optimizing the feature set, we were able to increase the mean score to 0.6484 of T+S F1 measure.

## 6 Conclusion

The machine learning approach with combination of methods for the distributional semantics (Word2Vec and GloVe) proved to be very capable of solving the advanced task of Interpretable Semantic Textual Similarity. We have chosen not to tune the system for individual datasets but to tune it for the task as a whole. The modified lexical semantic vectors approach seems to be an attractive alternative to the more traditional vector composition.

## Acknowledgments

This publication was supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports and by Grant No. SGS-2016-018 Data and Software Engineering for Advanced Applications. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures".

## References

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, Denver, Colorado, June. Association for Computational Linguistics.

Rajendra Banjade, Nobal Bikram Niraula, Nabin Maharjan, Vasile Rus, Dan Stefanescu, Mihai Lintean, and Dipesh Gautam. 2015. Nerosim: A system for measuring and interpreting semantic textual similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 164–171, Denver, Colorado, June. Association for Computational Linguistics.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March.

Leo Breiman, Jerome Friedman, Richard A. Olshen, and Charles Stone. 1984. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June.

Tomáš Brychcín and Lukáš Svoboda. 2016. Uwb at semeval-2016 task 1: Semantic textual similarity using lexical, syntactic, and semantic information. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, California, June.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, December.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. *SIGKDD Explorations Newsletter*, 11(1):10–18, November.

Christian Hänig, Robert Remus, and Xose de la Puente. 2015. Exb themis: Extensive feature extraction from word alignments for semantic textual similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 264–268, Denver, Colorado, June. Association for Computational Linguistics.

Sakethram Karumuri, Viswanadh Kumar Reddy Vuggumudi, and Sai Charan Raj Chitirala. 2015. Umduluthblueteam: Svcsts - a multilingual and chunk level semantic similarity system. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 107–110, Denver, Colorado, June. Association for Computational Linguistics.

Michal Konkol. 2014. Brainy: A machine learning library. In Leszek Rutkowski, Marcin Korytkowski, Rafa Scherer, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *Artificial Intelligence and Soft Computing*, volume 8468 of *Lecture Notes in Computer Science*, pages 490–499. Springer International Publishing.

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 Trans. on Knowl. and Data Eng.*, 18(8):1138–1150, August.

Dekang Lin. 1998. Extracting collocations from text corpora. In *First Workshop on Computational Terminology*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. 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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.